République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université SAAD DAHLEB de DLIDA

Faculté des Sciences

Département d'Informatique

Mémoire de fin d'études

En vue de l'obtention du diplôme de master en Informatique

Thème

Les systèmes de détection d'intrusions coopératifs avec la logique de description

Promotrice :

Mlle. Boustia

Réalisé par :

M. BOUMAZA Abdelkrim
M. NACEUR Mohamed Djihad

PROMOTION/2013

président de jury :
- Cherif-Zahar A
- cherFa
- Farah

ملخص

ان كشف التسلل كان محور العديد من الدراسات خلال السنوات الأخيرة اذ نظام كشف التسلل (IDS) هو عادة ما يعتبر خط الدفاع الثاني للحماية ضد الأنشطة الضارة وهذا بعد آليات أمنية مثل المصادقة والتحكم في الدخول.

IDSالتقليدية لديها نقاط ضعف،مدفوعة من هذه الملاحظة ووضع نظام للتعاون بين IDS والأدوات الأمنية الأخرى على أساس نظرية لجمع البيانات في المعرفية التعاونية المقترحة. هذا النظام يسمح للمقارنة بين حالة التأهب التي تم إنشاؤها بواسطة IDS والتحذيرات الواردة في قاعدة المعرفة التعاونية للاستدلال عما إذا كان هذا التنبيه هجوم أم لا. من خلال هذا النظام نستطيع ليس فقط أن نقارن بين التنبيهات ولكن يمكننا أيضا التفريق بين الإنذارات الصائبة والخاطئة.

Résumé

Durant ces dernières décennies, La détection d'intrusion fait l'objet de plusieurs études. Les systèmes de détection d'intrusion (IDS) sont habituellement considérés comme une deuxième ligne de défense pour se protéger contre les activités malicieuses, après d'autres mécanismes de sécurité tels que : l'authentification et le contrôle d'accès.

Les IDS traditionnels présentent des faiblesses, motivés par cette observation, un système de coopération entre les IDS et autres outils de sécurité en se basant sur la théorie de la collecte de données dans une base de connaissance coopérative a été proposé. Ce système permet une comparaison entre l'alerte généré par l'un des IDS et les alertes contenue dans la base de connaissance coopérative pour déduire si cette alerte représente une attaque ou non. Par ce système nous pouvons non seulement faire une comparaison entre les alertes, mais aussi différencier entre les vraies et les fausses alertes.

L'implémentation d'un système de détection d'intrusion coopérative contient : une base de connaissances coopérative, des scanneurs de vulnérabilités ainsi que plusieurs systèmes de détection d'intrusion avec différentes méthodes de détection. Ce système permettra aussi de diminuer les fausses alertes par des déductions faites à partir des informations récoltés des alertes générées par les IDS ainsi que d'autres outils de détection. Le résultat obtenu a montré que le système proposé est simple et efficace, vu que le taux de fausses alertes a diminué.

Abstract

In recent decades, Intrusion detection is the subject of several studies. The intrusion detection systems (IDS) are usually considered as a second line of defense to protect against malicious activities after other security mechanisms such as authentication and access control.

Traditional IDS have weaknesses, motivated by this observation, a system of cooperation between the IDS and other security tools based on the theory of collecting data in a cooperative knowledge was proposed. This system allows a comparison between the alert generated by an IDS and alerts contained in the cooperative knowledge base to infer if this alert is an attack or not. Through this system we can not only make a comparison between alerts, but also differentiate between true and false alarms.

The implementation of a system of cooperative intrusion detection includes: a base of cooperative knowledge, vulnerability scanners and many intrusion detection systems with different detection methods. This system will also reduce false alarms by deductions from information collected alerts generated by IDS and other detection tools. The result showed that the proposed system is simple and effective, as the false alarm rate decreased.

REMERCIEMENTS

Tout d'abord on tient à remercier notre DIEU le tout miséricordieux de nous avoir permis d'achever ce travail.

Nous adressons nos vifs remerciements à Mlle Boustia , notre promotrice , pour ses conseils, sa collaboration et ses encouragements.

Nous remercions notamment Lan Dickinson et Jushua Taylor qui nous ont aidé dans notre travail, et tous nos chers enseignants pour leurs efforts durant les 5 années d'étude.

Enfin, on remercie toute personne ayant contribuée de prés ou de loin à la progression de ce projet.

DEDICACES

Je dédie mon travail

A mes très chers parents qui ont toujours été là pour moi, et qui m'ont donné un magnifique modèle de labeur et de persévérance. J'espère qu'ils trouveront dans ce travail toute ma reconnaissance et tout mon amour.

A mon frère aîné Abdou, mes sœurs Sara, Oumaima ,Hiba et à toute ma famille.

A tous mes amis(es) et toute personne qui fait parti de mon entourage.

DEDICACES

Le premier mérite revient à mes très chers parents pour leur

grande patience, leur compréhension et leur soutien

Tous les mots ne sauraient témoigner de ma gratitude,

De mon amour, de mon respect, et de ma reconnaissance...

Que dieu leur procure bonne santé et longue vie

A ma très chère sœur Ryme et A mes chers frères

Mustapha et Nabil pour tout leur encouragement

A toute ma famille

A tous mes amis

Ainsi qu'à tous ceux qui me sont chers

TABLE DES MATIERES

TABLE DES MATIERES

CHAPITRE IV Réalisation du système.

TABLE DES FIGURES

LISTE DES TABLEUAX

1. INTRODUCTION

Les systèmes informatiques et les réseaux sont devenus des outils indispensables pour la société actuelle. Ils sont aujourd'hui déployés dans tous les secteurs professionnels. Initialement, isolés les uns des autres, ces systèmes informatiques sont devenus interconnectés et le nombre de points d'accès ne cesse de croître.

Ce développement phénoménal est accompagné également par la croissance du nombre d'utilisateurs, qui ne sont pas forcément pleins de bonnes intentions vis-à-vis de ces systèmes informatiques. Ils peuvent exploiter les vulnérabilités des réseaux et les systèmes pour essayer d'accéder à des informations sensibles dans le but de les lire, les modifier ou les détruire, portant atteinte au bon fonctionnement du système.

L'importance de sécurité des systèmes informatiques motive les angles divers de la recherche dont le but principal est de fournir de nouvelles solutions prometteuses qui ne pourraient être assurées par des méthodes classiques. Les systèmes de détection d'intrusions sont l'une de ces solutions qui permettent la détection des utilisations non autorisées, les mauvaises utilisations et les abus dans un système informatique par les utilisateurs externes ainsi que ses utilisateurs internes.

Le défi dans le domaine de la sécurité informatique et plus précisément dans les systèmes de détection d'intrusions est de pouvoir déterminer la différence entre un fonctionnement normal et un fonctionnement avec intrus.

A cet effet, plusieurs approches on été proposées dans la littérature, l'inconvénient de ces approches est qu'elles produisent plusieurs fausses alertes. Afin de gérer ce problème, une approche prometteuse a été récemment proposée. Cette approche appelée coopérative permet à plusieurs systèmes de détection d'intrusions (IDS) de coopérer entre eux.

La contribution de ce travail s'est fixé comme objectif, une étude approfondie des systèmes de détection d'intrusions, et une compréhension de la logique de description (DL) pour permettre de suggérer une proposition d'un système coopératif de détection

d'intrusions avec la logique de description en utilisant des systèmes de détection d'intrusion avec des méthodes de détection différentes.

Pour ce faire, ce mémoire est scindé en deux grandes parties. Une première partie bibliographique dans laquelle le chapitre 1 est consacré à l'étude des systèmes de détection d'intrusion en détaillant leurs méthodes de détection et leur classification. Le second chapitre, aborde l'étude de la logique de description. Enfin, le dernier chapitre décrit les efforts de standardisation faits dans le domaine de l'interopérabilité entre plusieurs systèmes de détection d'intrusions.

La deuxième partie de ce travail est particulièrement consacrée à la présentation d'un formalisme logique de représentation des informations contextuelles en détection d'intrusions, à l'approche proposée pour la coopération des IDS ainsi qu'à la diminution des fausses alertes et ce, dans le chapitre 4. Quant au chapitre 5, il est consacré à la présentation de la réalisation du système coopératif et aux explications nécessaires pour la mise en œuvre de notre application.

Chapitre I :

LES SYSTEME DE DETECTION D'INTRUSIONS

1. Introduction

Avec le développement des réseaux de communication, l'Internet est devenu l'infrastructure critique pour une société moderne. La croissance explosive d'utilisateurs d'Internet a motivé l'expansion rapide de commerce électronique et d'autres services en ligne. Malheureusement derrière la convenance et l'efficacité de ces services, les risques et les chances d'intrusions malveillantes sont aussi augmentés. La sécurité des systèmes informatiques est devenue un défi majeur dont l'objectif est d'assurer la disponibilité des services, la confidentialité et l'intégrité des données et des échanges.

De nombreux mécanismes ont été développés pour assurer la sécurité des systèmes informatiques, particulièrement pour *prévenir les intrusions* dont le but principal est de construire un système sécurisé en déterminant et éliminant les vulnérabilités de sécurité. Citons par exemple l'authentification qui consiste à prouver l'identité des utilisateurs, le contrôle d'accès qui consiste à définir les droits d'accès accordés aux utilisateurs sur les données et les pare-feux qui filtrent l'accès aux services du système informatique vis-à-vis de l'extérieur.

Cependant, ces mécanismes ne peuvent pas garantir la sécurité des systèmes informatiques. En effet, ces systèmes présentent des failles de conception, d'implémentation et de configuration permettant à des attaquants de contourner les mécanismes de prévention. De plus, un certain nombre de ces systèmes se protègent contre des attaques externes, alors plusieurs études ont montré qu'il existe des attaques issues de leurs utilisateurs internes.

L'augmentation de l'importance de sécurité d'ordinateurs mène à des recherches diverses afin de fournir de nouvelles solutions qui ne pourraient pas être réalisables par des approches de sécurité classiques. Les systèmes de détection d'intrusions sont l'une de ces solutions qui permettent de garantir la sécurité.

Pour cette raison, ce chapitre sera consacré à présenter cette nouvelle solution destinée à repérer des activités anormales ou suspectes sur la cible analysée qui peut être un hôte ou un réseau par exemple. Ce chapitre sera organisé comme

suit : nous commencerons d'abords par la description d'un système de détection d'intrusions (IDS), ainsi nous présentons les critères nécessaires pour assurer l'efficacité de tel système. La section suivante sera consacrée à la classification des systèmes de détection d'intrusions selon plusieurs critères. Puis, nous étudierons quelques types d'IDS d'une manière détaillée vu leur importance dans le reste de cette étude. Finalement, nous exposerons d'une manière générale les architectures d'implémentation des systèmes de détection d'intrusions ainsi que la présentation de quelques systèmes de détection d'intrusions existants.

2. Le système de détection d'intrusions

La recherche dans le domaine de système de détection d'intrusions a commencée en 1980 par le travail de James Anderson [1] qui a souhaité l'amélioration des équipements d'audit et les capacités de surveillance des systèmes informatiques. Suite à ce travail, le premier modèle de détection d'intrusions générique a été proposé par Dorothy Denning en 1987 [2]. Ce modèle de détection d'intrusions générique est indépendant de tout système et environnement d'applications, des types d'intrusions et les vulnérabilités de système [2]. IL se compose de six éléments:

- ***Les sujets*** : ce sont les initiateurs des actions effectuées sur le système qui peuvent être : les utilisateurs, les groupes d'utilisateurs, ou encore les processus générés par ces utilisateurs.
- ***Les objets*** : ce sont les ressources gérées par le système tels que les fichiers, les programmes, les messages, les périphériques, etc.
- ***Les enregistrements d'audit*** : ils représentent les actions entreprises par un sujet sur un objet et sont générés par le système. Ces enregistrements sont composés par :
  - Le sujet ayant fait l'action.
  - Le type de l'action (par exemple login, logout, lecture, écriture).
  - Les ressources utilisées.

- Le résultat de l'action (échec ou succès).
- Des éléments quantitatifs (par exemple, nombre de lignes ou pages imprimées, nombre d'enregistrements lus ou écrits).
- La date et l'heure où s'est déroulée l'action.

➢ ***Les profils*** : ce sont des structures qui caractérisent le comportement des sujets envers des objets.

➢ ***Les enregistrements d'anomalie*** : ils sont générés lorsqu'une activité anormale est détectée.

➢ ***les règles d'activités*** : elles spécifient les actions à entreprendre lorsque certaines conditions sont satisfaites sur les enregistrements d'audit ou les enregistrements d'anomalies générés.

Dans ce modèle, un profil est défini par un ensemble de *variables* et de *modèles statistiques*. Ces variables représentent des mesures quantitatives accumulées durant une période de temps, qui peut être soit un intervalle de temps fixe (heure, jour, etc.) ou entre deux événements d'audits (login et logout, connexion et déconnexion, etc.). Ce modèle de base est un système expert en temps réel. Il stocke les faits des profils décrivant le comportement normal afin de générer les signatures de comportements incorrects. Quand un sujet agit sur un objet spécifique alors l'événement généré change la valeur des variables. Une base de connaissances contient les règles d'activité dont le rôle de ces règles consiste à mettre à jour les profils, détecter les comportements anormaux, produire des rapports ou encore alerter l'officier de sécurité. Le moteur d'inférence essaye de déclencher les règles qui correspondent aux faits des profils. En ce qui concerne les modèles statistiques proposés par Denning, ils seront décrits dans les sections suivantes.

2.1. Définition d'un système de détection d'intrusions

Il existe plusieurs définitions du système de détection d'intrusions.

Cependant, nous allons seulement citer quelques unes.

2.1.1 *Définition 1*

Heady et d'autres [3] ont défini un système de détection d'intrusions comme suit :

Définition 1 : Intrusion

Une intrusion est n'importe quel ensemble des actions qui essayent de compromettre l'intégrité, la confidentialité ou la disponibilité d'une ressource d'ordinateur.

Définition 2 : Détection d'intrusions

C'est le problème d'identification des actions qui essayent de compromettre l'intégrité, la confidentialité ou la disponibilité d'une ressource d'ordinateur.

Définition 3 : Système de détection d'intrusions

C'est un système d'ordinateur (une combinaison de logiciels et de matériels) qui essaye d'exécuter la détection d'intrusions.

2.1.2 *Définition 2*

Tandis que Kim [4] a défini un système de détection d'intrusions comme un système automatisé dont le rôle est la détection des intrusions dans un système informatique tout en examinant les audits de sécurité fournis par le système d'exploitation ou bien les outils de contrôle du réseau. Son but principal est la détection des utilisations non autorisées, les mauvaises utilisations et les abus dans un système informatique par les utilisateurs internes et externes.

2.1.3 *Définition 3*

La détection d'intrusions consiste à surveiller des événements qui se produisent dans un ordinateur ou dans un réseau informatique et de les analyser pour découvrir des signes d'intrusions. Ces événements sont souvent définis comme des tentatives de violation de la politique de sécurité. Les intrusions ont plusieurs causes tels que : des logiciels malveillants (par exemple, virus, cheval de Troie, etc.), l'accès des attaquants externes aux systèmes via des réseaux ouverts comme Internet, des utilisateurs autorisés qui essayent de gagner des privilèges additionnels pour lesquels ils ne sont pas autorisés, ou des utilisateurs autorisés qui abusent de leurs

privilèges [5].

2.2. Terminologie des attaques

Une attaque peut être définie comme toute action ou ensemble d'action qui peut porter atteinte à la sécurité des informations d'un système ou d'un réseau informatique.

2.2.1. *Classification des attaques*

Vu le nombre important des attaques possibles, elles peuvent être classées selon différentes classifications [6] :

2.2.1.1 La premiere classification

- ***Les attaques passives :*** ce type d'attaque vise à l'obtention d'accès pour pénétrer dans le système sans compromettre ces ressources.
- ***Les attaques actives :*** le résultat de cette attaque est un changement non autorisé d'état des ressources système.

2.2.1.2 La deuxième classification

- ***Les attaques internes*** : ce type d'attaque est causé :

  Soit par les utilisateurs autorisés du système qui essayent d'utiliser des privilèges complémentaires dont ils n'ont pas le droit.

  Soit par les utilisateurs autorisés qui emploient improprement les privilèges dont ils ont le droit.

- ***Les attaques externes*** : ce type d'attaque est causé par des utilisateurs externes qui essayent d'accéder à des informations ou des ressources d'une manière illégitime et non autorisée.

2.2.1.3 La troisième classification

Selon cette classification, les attaques de cette catégorie peuvent porter atteinte à :

- *La confidentialité* des informations en brisant les règles privées.
- *L'intégrité* en altérant les données.

- *La disponibilité* en rendant un système ou un réseau informatique indisponible. Ces attaques sont connues sous le nom des attaques de déni de service.
- *L'authenticité* des informations.

2.2.2. *Description des attaques*

Il existe différentes attaques que les systèmes de détection d'intrusions essayent de repérer. Parmi ces attaques, nous citons par exemple [6] :

- ***Craquage des mots de passe.***
- ***Cheval de Troie « Trojan Horse »*** qui est un programme qui se cache lui-même dans un autre programme au-dessus de tout soupçon. Quand la victime lance ce programme, elle lance aussi le cheval de Troie caché. Un cheval de Troie peut par exemple, lorsqu'il est exécuté, ouvrir l'accès au système à des personnes particulières ou même à tout le monde.
- ***IP spoofing*** : dans ce cas l'attaquant change son adresse IP par une autre adresse de confiance afin d'obtenir des droits d'accès.
- ***Les scans*** : qui servent principalement à obtenir des informations sur un hôte, un réseau (pour préparer une attaque plus élaborée). Les informations qu'un attaquant peut obtenir sur le réseau sont par exemple : le type de système d'exploitation de la machine, les ports ouverts, etc.
- ***Sniffing:*** elle se caractérise par l'observation et l'analyse du trafic réseau afin d'obtenir des informations pertinentes pour les attaques suivantes.
- ***Les attaques de déni de service (denial of service)*** ont pour but de paralyser le serveur cible pour qu'il devienne inaccessible, au moins pour une durée de temps. De très nombreuses techniques existent pour épuiser les ressources d'un hôte cible, par exemple : ICMP Flooding, smurf, SYN flood, etc.
- Etc.

2.3. Concepts de base

Nous désirons dans cette section d'éclairer quelques notions qui seront utilisées dans le reste de ce travail.

- **Système** : dénote un système d'information contrôlé par un système de détection d'intrusions. Cela peut être un poste de travail, un élément du réseau, une unité centrale, un pare-feu, un serveur Web, un réseau d'entreprise, etc.
- **Alarme** : c'est la réponse générée par le système de détection d'intrusions lors de la détection d'une intrusion. Cependant les erreurs de détection peuvent être classées selon deux types :

  Le positif faux : signifie qu'un système de détection d'intrusions détecte une intrusion là où aucune intrusion réelle n'a été commise.

  Le négatif faux : A l'inverse de «positif faux », «négatif faux » signifie que le système de détection d'intrusions n'a pas détecté une intrusion ayant réussi.

2.4. Description du système de détection d'intrusions

Un système de détection d'intrusions à un niveau très macroscopique [7] peut être décrit comme un *détecteur*. Ce détecteur est un moteur d'analyse qui reçoit des données de trois sortes de ressources (Figure 1.1). L'analyse de ces données génère une décision d'évaluation de la probabilité que ces actions peuvent être considérées comme des symptômes d'intrusions. Ces données sont:

- Des informations de configuration relatives à l'état actuel du système.
- Des informations à long terme relatives à la technique utilisée pour détecter les intrusions par exemple une base de connaissances d'attaques.
- Des informations venant du système à protéger qui sont les informations d'audit décrivant les événements qui apparaissent dans le système.

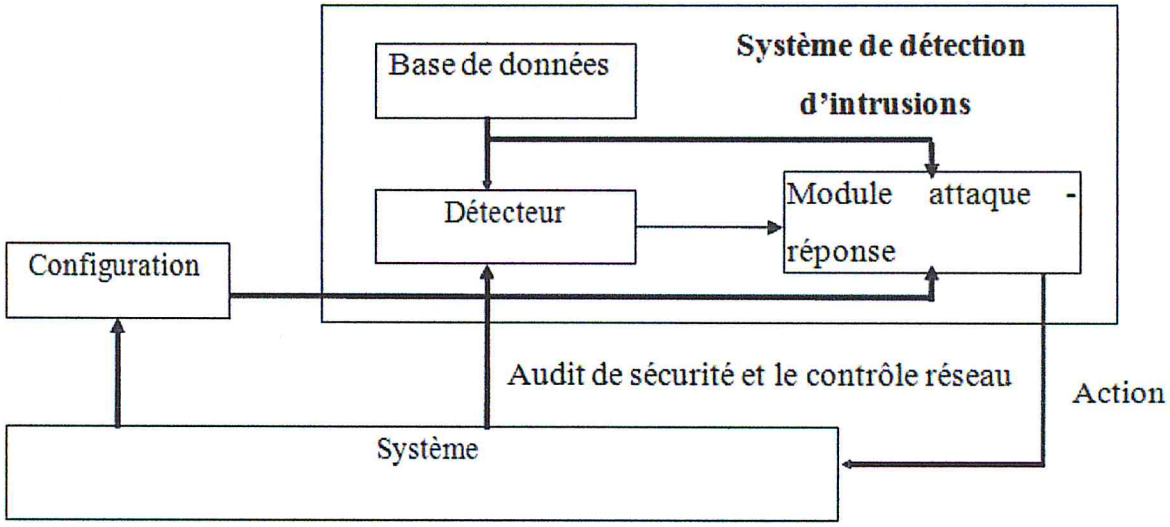


Figure 1.1 : Description d'un système de détection d'intrusions.

3. Efficacité des systèmes de détection d'intrusions

L'efficacité d'un système de détection d'intrusions est déterminée par les mesures suivantes [7] :

- **Exactitude** : Le système de détection d'intrusions n'est pas exact s'il considère les actions légitimes des utilisateurs comme atypiques ou intrusives.
- **Performance :** La performance d'un système de détection d'intrusions est mesurée par le taux de traitement des traces d'audits. Si la performance du système de détection d'intrusions est pauvre, donc la détection en temps réel n'est pas possible.
- **Perfection :** Un système de détection d'intrusions est imparfait s'il n'arrive pas à détecter une attaque.
- **Tolérance aux pannes :** Un système de détection d'intrusions doit être résistant aux attaques, en particulier dans le cas des attaques de déni de service.
- **Opportunité :** Un système de détection d'intrusions doit exécuter et propager son analyse d'une manière prompte pour permettre une réaction rapide dans le cas d'existence d'une attaque.

4. Classification des systèmes de détection d'intrusions

4.1. Une vue d'ensemble du système IDS

Les différents systèmes de détection d'intrusions disponibles peuvent être classés [7,8] selon plusieurs critères (Figure 1.2), qui sont :

- La méthode de détection.
- Le comportement du système après la détection.
- La source des données.
- La fréquence d'utilisation.

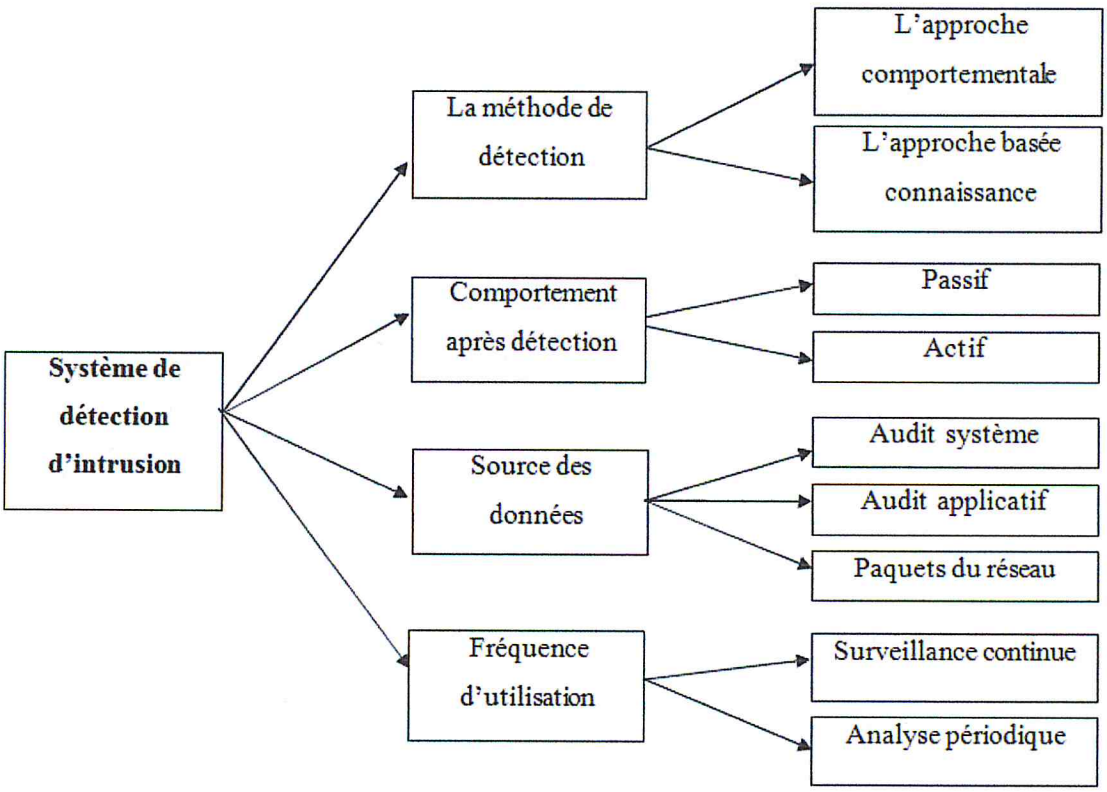


Figure 1.2 : Classification d'un système de détection d'intrusions

4.2. La méthode de détection

La détection d'intrusions repose sur deux approches de base :

- L'approche comportementale.
- L'approche basée connaissance.

4.2.1. *L'approche comportementale*

Cette approche est connue aussi par ***l'approche de détection d'anomalies.*** Elle consiste à définir un profil de l'activité normale d'un utilisateur et à considérer les déviations significatives de l'activité d'utilisateur courante par rapport aux profils de comportement normaux comme anomalie.

4.2.2. *L'approche basée connaissance*

Cette approche définit des signatures soupçonneuses basées sur les vulnérabilités connues de système et la politique de sécurité. Une intrusion est signalée lorsque la trace d'une attaque connue est présente dans les traces d'audit.

Ces deux méthodes d'analyse constituent la partie importante des systèmes de détection d'intrusions. Pour cette raison, elles seront détaillées dans les sections suivantes.

4.3. Le comportement après la détection d'intrusions

Le comportement d'un IDS après la détection d'une intrusion est l'ensemble des actions prises par le système lorsqu'il détecte une attaque. Ces réponses peuvent être *actives* ou bien *passives.*

4.3.1. *Réponse active*

La réponse active implique des actions automatisées prises par un IDS quand le système détecte une intrusion. Par exemple interrompre le progrès d'une attaque pour bloquer ensuite l'accès suivant de l'attaquant.

4.3.2. *Réponse passive*

Dans ce cas, quand une attaque est détectée, le système de détection d'intrusions

ne prend aucune action. Il génère seulement une alarme pour notifier l'administrateur de système qui va prendre des mesures en se basant sur les rapports générés par le système de détection d'intrusions.

4.4. La nature des données analysées

Les systèmes de détection d'intrusions sont classés en fonction de l'origine des données qui seront exploitées pour détecter des actions intrusives. La source de données utilisée est une caractéristique essentielle pour classer les systèmes de détection d'intrusions. On distingue trois catégories de sources d'informations:

- Les audits systèmes.
- Les audits applicatifs.
- Le trafic réseau.

4.4.1. *Les audits systèmes*

Les audits systèmes sont produits par le système d'exploitation d'un hôte. Ces données permettent à un IDS de contrôler les activités d'un utilisateur sur un hôte. Elles peuvent être également de plusieurs types, par exemple :

- ***Historique des commandes systèmes :*** tous les systèmes d'exploitation possèdent des commandes pour obtenir *des informations instantanées* sur les processus actifs courants dans un ordinateur. Grâce à ces commandes, l'IDS peut avoir des informations précises sur les événements systèmes.
- ***Accounting :*** l'accounting fournit des informations sur l'usage des ressources partagées par les utilisateurs. Ces ressources sont par exemple : le temps processeur, la mémoire, l'espace disque, les applications lancées, etc.
- ***Systèmes d'audit de sécurité :*** les systèmes d'exploitation sont dotés par ce service pour définir des événements, les associer à des utilisateurs et assurer leurs collectes dans un fichier d'audit. L'IDS possède potentiellement des informations sur toutes les actions effectuées par un utilisateur.

L'avantage de ces données systèmes réside dans leur fiabilité et leur granularité fine, qui permettent un diagnostic précis des actions effectuées sur un hôte par

un attaquant. Cependant, le volume d'événements généré par les audits systèmes est très volumineux ce qui implique un impact très important sur les performances de la machine surveillée.

Les IDS qui se basent sur cette catégorie des sources de données sont appelés : **Les IDS basés hôte « Host Based Intrusion Detection System ».**

4.4.2. *Les sources d'informations réseau*

Ce sont des données du trafic réseau. Cette source d'informations est prometteuse car elle permet de collecter et analyser les paquets de données circulant sur le réseau.

Les IDS qui exploitent ces sources de données sont appelés : **Les IDS basés réseau « Network Based Intrusion Detection System ».**

4.4.3. *Les audits applicatifs*

La troisième catégorie de source de données est constituée des audits applicatifs. Les données à analyser sont produites directement par une application, par exemple des fichiers logs générés par les serveurs ftp et les serveurs Web. L'avantage de cette catégorie est que les données produites sont très synthétiques, elles sont sémantiquement riches et leur volume est modéré. On note que ces types d'informations sont généralement intégrés dans les IDS basés hôte.

Vu l'importance des IDS basés hôte et basés réseau, une étude détaillée de ces deux types d'IDS sera présentée dans les prochaines sections.

4.5. La fréquence d'utilisation

La fréquence d'utilisation d'un système de détection d'intrusions peut exister selon deux formes :

4.5.1. *Surveillance périodique*

Ce type de système de détection d'intrusions analyse périodiquement les différentes sources de données à la recherche d'une éventuelle intrusion ou une

anomalie passée.

4.5.2. *Surveillance en temps réel*

Les systèmes de détection d'intrusions en temps réel fonctionnent sur le traitement et l'analyse continue des informations produites par les différentes sources de données. La détection d'intrusions en temps réel permet de limiter les dégâts produits par une attaque car elle permet de prendre des mesures qui réduisent le progrès de l'attaque détectée.

4.6. Critères d'évaluation d'un IDS

Définir des critères quantitatifs d'évaluation est d'une importance capitale pour les utilisateurs potentiels des résultats des évaluations des IDS tels que les opérateurs de sécurité et les chercheurs. Les opérateurs de sécurité ont besoin de tels critères pour améliorer le processus de sélection des IDS, qui est trop souvent basé uniquement sur les revendications des constructeurs et les commentaires portés dans des revues spécialisées.

Par exemple, les opérateurs de sécurité qui examinent les Logs des IDS voudraient savoir la probabilité que des alertes se produisent lorsque certains types d'attaques sont lancés. Les chercheurs également ont besoin de ces évaluations pour identifier les forces et les faiblesses des systèmes actuels afin de concentrer les efforts de recherche sur l'amélioration de ces systèmes.

Dans [9], les auteurs reprennent les critères initialement proposés dans [10] et [11] et rajoutent d'autres critères pour évaluer l'efficacité des systèmes de détection d'intrusions :

- Couverture : ce critère mesure le nombre d'attaques qui peuvent être détectées par l'IDS en mode de fonctionnement idéal. Pour un IDS basé signatures, ce critère est déterminé par le nombre de signatures définies dans le IDS.
- Taux de fausses alertes : ce critère mesure le taux de fausses alertes générées par un IDS dans un environnement donné et pendant une durée donnée. On

désigne par faux positif toute alerte qui correspond à une activité légale. Les causes des fausses alertes sont diverse. Certaines fausses alertes sont dues à un trafic légitime généré par des outils d'administration. Ce critère est difficile à évaluer car il diffère d'un environnement de test à un autre.

- Taux de détection : ce critère mesure le taux des attaques détectées par un IDS dans un environnement donné et pendant une durée donnée. On désigne par faut négatif toute attaque ratée par le IDS. Ce critère est aussi difficile à évaluer, parce qu'il est impossible d'avoir une connaissance globale sur toutes les attaques.
- Résistance aux attaques : ce critère mesure comment un IDS résiste aux attaques qui ciblent à perturber son fonctionnement. Un exemple d'attaque consiste à envoyer une grande quantité de trafic qui ne contient pas d'attaques, dans le but de dépasser les capacités de traitement de l'IDS. Ainsi, les vraies attaques qui se lancent au moment de la surcharge du IDS peuvent passer inaperçues.
- Capacité de travailler en haut-débit : ce critère montre à quel point un IDS fonctionne normalement lorsqu'il confronte un grand volume de trafic. La plupart des IDS réseaux commencent à perdre des paquets lorsque le volume du trafic augmente, ce qui causera à l'IDS de ne pas détecter un certain nombre d'attaques.
- Capacité de corréler des événements : ce critère montre à quel point un IDS peut corréler les événements causés par les attaques. Ces événements peuvent être recueillis auprès des IDS, des routeurs, des pare-feux, des journaux d'applications, etc.

  Un des principaux objectifs de cette corrélation est d'identifier les scénarios d'attaque en cours.
- Capacité de détecter les nouvelles attaques : ce critère montre à quel point un IDS peut détecter de nouvelles attaques. Il n'est généralement pas utile de mesurer ce critère pour les IDS à base de signatures car ils ne peuvent détecter que les attaques connues. Toutefois, les systèmes basés sur la détection des anomalies peuvent être appropriés pour ce type de mesures. Généralement, les

IDS ayant cette capacité de détecter les nouvelles attaques produisent davantage de faux positifs que ceux qui n'ont pas cette fonctionnalité.

- Capacité d'identifier les attaques correctement : ce critère montre à quel point un IDS peut identifier les attaques détectées et leur assigner les noms d'attaque/ vulnérabilité les plus en communs et les classer dans les bonnes catégories.
- Capacité d'identifier la réussite des attaques : ce critère montre si l'IDS peut déterminer la réussite des attaques à partir des systèmes attaqués. Dans les environnements réseaux actuels, plusieurs attaques visant la remontée en privilèges à distance échouent et n'endommagent pas les systèmes attaqués. Cependant, beaucoup d'IDS ne distinguent pas entre l'échec et la réussite de ces attaques. La capacité de déterminer la réussite des attaques est essentielle pour l'analyse et la corrélation des attaques. Elle simplifie aussi le travail de l'opérateur de sécurité en distinguant entre les attaques réussies qui sont très importantes et les tentatives d'attaques échouées qui sont généralement moins importantes.

5. L'analyse basée connaissance versus l'analyse comportementale

Comme nous l'avons vu dans la section précédente, il existe deux approches pour détecter les intrusions dans les systèmes informatiques [7, 8,12]. L'approche *basée connaissance* qui se base sur la définition d'un modèle constitué des actions interdites dans les systèmes informatiques et *l'approche comportementale* qui est basée sur la définition d'un modèle constitué des actions autorisées.

5.1. L'analyse basée connaissance

Cette approche de détection est désignée en anglais par le terme « ***Misuse Detection*** », qui signifie dans la littérature la détection d'une mauvaise utilisation, et il existe plusieurs traductions françaises adoptées pour cette approche, par exemple l'approche par signatures ou par scénarios.

Elle est caractérisée par l'existence d'une base de connaissances qui comporte des modèles d'attaque connus a priori qui sont appelés *les signatures*. Elle

examine les activités du système et du réseau en cherchant des événements ou l'ensemble des événements qui décrivent une attaque connue. Ainsi, dans cette approche, tout ce qui n'est pas explicitement interdit est autorisé. Cette approche possède un certain nombre d'avantages et d'inconvénients [8, 13].

5.1.1. *Avantages*

- L'analyse basée connaissance est très efficace pour la détection d'attaque avec un taux très bas des alarmes de type positif faux.
- Les alarmes générées sont significatives.

5.1.2. *Inconvénients*

- Cette analyse basée connaissance permet seulement la détection des attaques qui sont connues au préalable. Donc, la base de connaissances doit être constamment mise à jour avec les signatures de nouvelles attaques.
- Le risque que l'attaquant peut influencer sur la détection après la reconnaissance des signatures.

5.2. L'analyse comportementale

Dans l'analyse comportementale, un modèle de comportement normal du système surveillé est préalablement construit. Ce modèle est appelé *profil de comportement normal qui* sera utilisé comme une référence dans la détection. Au cours de la surveillance du système, toute déviation significative du comportement courant de système contrôlé par rapport au comportement normal de référence donne lieu à une attaque. Cette approche possède aussi un certain nombre d'avantages et d'inconvénients [8, 13].

5.2.1. *Avantages*

- L'analyse comportementale n'exige pas des connaissances préalables sur les attaques.
- Elle permet la détection de la mauvaise utilisation des privilèges.
- Elle permet de produire des informations qui peuvent être employées pour définir des signatures pour l'analyse basée connaissance.

5.2.2. *Inconvénients*

- Les approches comportementales produisent un taux élevé des alarmes de type positif faux en raison des comportements imprévisibles d'utilisateurs et des réseaux.
- Ces approches nécessitent des phases d'apprentissage pour caractériser les profils de comportement normaux.
- Les alarmes générées par cette approche ne sont pas significatives.

Cette étude comparative entre les deux approches d'analyse utilisées par les systèmes de détection d'intrusions montre l'existence d'une complétude entre ces deux méthodes. Cette complétude qui permettra de surmonter les inconvénients relatifs à chaque méthode d'analyse. Pour cette raison, il est préférable d'adopter les deux techniques d'une manière parallèle pour obtenir un système de détection d'intrusions efficace [4]. Cependant, les systèmes de détection d'intrusions commerciaux disponibles emploient seulement la technique basée signature, ce qui motive les efforts de recherche croissants pour construire des détecteurs d'anomalies efficaces pour des buts de détection d'intrusions. L'effort principal de cette recherche est concentré sur les systèmes de détection d'intrusions qui sont basés sur la technique comportementale.

Pour cette raison, nous présenterons dans la section suivante les différentes approches utilisées dans la méthode de détection comportementale.

6. Les techniques utilisées dans l'approche comportementale

Un système de détection d'intrusions basé sur la détection d'anomalies contrôle les activités du système afin de les classer comme normales ou anomalies. Il procède par la construction des profils d'un comportement normal pour les activités des utilisateurs et l'observation des déviations significatives de l'activité de l'utilisateur courante par rapport à la forme normale établie. D'une façon générale, la détection d'anomalies est composée de deux phases :

- **Une phase d'apprentissage** : le système apprend le comportement normal d'un utilisateur ou un système. Il crée ainsi ***« le profil normal »*** d'un utilisateur ou d'un système à partir des données collectées.
- **Une phase de détection** : le système compare les traces d'audit courantes ou le trafic réseau aux profils pour vérifier s'il n'y a pas une activité intrusive. Si la différence entre le profil et les traces d'audit est significative, une alarme est déclenchée.

Pour pouvoir formaliser le comportement normal d'un système, des approches diverses ont été utilisées [7, 12,4]. Cette section sera consacrée à une présentation générale de ces différentes approches.

6.1. L'approche statistique

L'approche statistique est utilisée pour la génération d'un modèle de comportement normal d'un système. Elle consiste à générer le profil de comportement normal à partir d'un ensemble de variables aléatoires, échantillonnées à des intervalles réguliers dans le temps, ces variables peuvent être par exemple :

- Le temps CPU utilisé.
- Le nombre de connexions établi durant une période de temps.
- Les fichiers les plus fréquemment utilisés.
- Les entrées/sorties effectuées.
- Etc.

Dans cette approche, Denning [2] a proposé un ensemble de modèles statistiques, leur but est de définir à partir de n observations $x_1, x_2, \ldots x_n$ sur une variable donnée

x, si la valeur x_{n+1} de l'observation n+1 est anormale. Parmi ces modèles, on peut citer les modèles suivants :

- ***Le modèle opérationnel*** : ce modèle est très simple, une anomalie est détectée par la comparaison de la valeur d'une nouvelle observation avec un seuil fixe qui est défini d'une manière intuitive en se basant sur les données historiques.
- ***Le modèle de déviation standard et moyen*** : Ce modèle définit un seuil d'anomalie par l'estimation d'un intervalle de confiance. L'intervalle de confiance est la moyenne et l'écart type des n observations qui peuvent être considérées normales. Si la valeur d'une nouvelle observation est en dehors de cet intervalle alors elle est considérée anormale.
- ***Le modèle de covariances*** : Il est similaire au modèle précédent mais il se base sur la corrélation de plusieurs variables pour tirer des conclusions.

Ces approches ont été adoptées dans le développement de plusieurs systèmes de détection d'intrusions, on peut citer par exemple :

- MIDAS « Multics Intrusion Detection and Alerting System » [14].
- NIDES « Next Generation Real time Intrusion Detection Expert System » [15].

6.2. L'approche de la machine learning

Le but principal de l'utilisation de la machine Learning est l'extraction automatique des caractéristiques des activités normales qui sont critiques pour la détection d'anomalies. A partir des données d'audit, le modèle de la machine Learning essaye d'identifier des règles pour définir les comportements normaux. Ces règles seront employées pour déterminer si des événements nouvellement observés sont anormaux ou non.

Parmi les travaux qui sont basés sur cette approche, le système de détection d'intrusions basé règle TIM « Time based Inductive Machine » [16] proposé par Teng et son groupe. TIM génère des règles qui essayent de prédire les événements futurs en se basant sur des événements qui se sont déjà produits dans

le passé.

Durant la phase d'apprentissage, cette approche détermine des règles temporelles qui caractérisent le comportement normal des utilisateurs. Les règles ont par exemple la forme suivante :

> E1 -> E2 -> E3 -> (E4 = 95%, E5 = 5%) où E1, E2,... E5 sont des événements.

Cette règle signifie que la séquence d'évènements observée E1-> E2 ->E3 implique ensuite l'occurrence de l'évènement E4 avec une probabilité de 95% ou l'occurrence de l'évènement E5 avec une probabilité de 5%.

Durant la phase de détection, les règles possédant des parties gauches qui correspondent à la séquence d'événements observés seront sélectionnées et l'évènement prédit de cette règle sera comparé avec le dernier évènement qui apparaît dans la séquence d'évènements observée. Si cet évènement dévie d'une manière significative de ceux prédits dans la règle, alors TIM alerte l'officier de sécurité.

6.3. L'approche de réseaux de neurones

Les réseaux de neurones sont utilisés dans la détection d'anomalies afin d'exploiter leurs capacités d'apprentissage. L'idée de base est d'utiliser les mécanismes d'apprentissage des réseaux de neurones pour apprendre les profils de comportements normaux des utilisateurs ou d'un système.

Plusieurs travaux ont été élaborés qui ont essayé d'abord d'apprendre à un réseau de neurones le comportement normal d'un système pour qu'il puisse par la suite décider si un ensemble d'action est normal ou suspect. Parmi ces travaux, nous citons le travail de Debar [17] qui a proposé l'utilisation des réseaux de neurones pour construire un modèle du comportement des utilisateurs du système informatique. Le travail proposé s'intéresse à l'aspect dynamique du comportement et à sa présentation sous des séries d'actions temporelles.

6.4. L'approche de data mining

Le but de cette approche est l'exploitation des techniques de data mining pour extraire des anomalies à partir des grandes quantités de données du trafic

réseau. Parmi les travaux existants, on peut citer ADAM « Audit Data Analysis and Mining » [18] qui est un système de détection d'intrusions qui exploite des techniques de data mining pour construire des profils du trafic réseau normaux. ADAM utilise les règles d'association pour construire des profils du trafic de réseau normaux qui seront employées par la suite pour détecter les comportements incorrects de trafic de réseau. Pour détecter des anomalies, ADAM extrait les règles d'association à partir des données du trafic réseau et qui seront comparées aux profils du réseau. Si n'importe quelle règle d'association produite à partir des données de trafic de réseau rassemblées n'est pas incluse dans les profils, alors cette règle est considérée comme une indication d'un comportement incorrect.

6.5. L'approche immunologique

Vu que la détection d'anomalies est une application directe de la métaphore immunitaire, plusieurs travaux tentent de calquer la manière dont le système immunitaire naturel procède pour la distinction entre le comportement normal et le comportement suspect afin de construire des systèmes de détection d'intrusions efficaces. Parmi ces travaux nous citons le système LYSIS [19,20] qui a intégré des différentes propriétés et mécanismes inspirés par le système immunitaire humain. Il se base principalement sur l'algorithme de la sélection négative proposé par Forrest [21]. Dans ce travail, une population de détecteurs est générée aléatoirement, puis en se basant sur les modèles de comportement normaux des utilisateurs, les détecteurs qui identifient ces modèles seront éliminés, en d'autre terme élimination des détecteurs qui détectent le soi. La population des détecteurs restants procède à contrôler les opérations effectuées dans le système de telle sorte que s'il y a une correspondance entre un détecteur et l'opération courante dans le système alors cette opération est considérée litigieuse ou anormale.

7. Les IDS basés hôtes versus les IDS basés réseau

En raison des multiples possibilités d'attaques des systèmes informatiques et des réseaux, il existe différents types de systèmes de détection d'intrusions [7, 8, 22,22] qui varient selon l'endroit qu'ils surveillent et ce qu'ils contrôlent (les sources d'information).

7.1. L'IDS basé hôte (host- based IDS)

L'IDS basé hôte contrôle un seul hôte. Il analyse des informations rassemblées d'un système d'ordinateur individuel, ce qui permet à l'IDS basé hôte d'analyser des activités avec une grande fiabilité et précision en déterminant exactement les processus et les utilisateurs impliqués dans une attaque particulière. Les avantages et les inconvénients [8, 13] de l'IDS basé hôte sont :

7.1.1. *Avantages*

- La capacité de contrôler les activités locales des utilisateurs avec précision.
- Capable de déterminer si une tentative d'attaque est couronnée de succès.
- La capacité de fonctionnement dans des environnements cryptés.
- L'IDS basé hôte fonctionne sur les traces d'audit des systèmes d'exploitation ce qui lui permet de détecter certains types d'attaques (ex : Cheval de Troie).

7.1.2. *Inconvénients*

- La vulnérabilité aux attaques du type déni de service puisque l'IDS peut résider dans l'hôte cible par les attaques.
- La difficulté de déploiement et de gestion, surtout lorsque le nombre d'hôtes qui ont besoin de protection est large.
- Ces systèmes sont incapables de détecter des attaques contre de multiples cibles dans le réseau.

7.2. L'IDS basé réseau (Network-

based IDS)

Bien que le système de détection d'intrusions basé hôte a montré des résultats encourageants, son problème majeur est la détection des intrusions fait à travers le réseau. Pour détecter cette sorte d'intrusion, l'IDS a besoin de contrôler des événements multiples produits sur plusieurs hôtes. En effet, une proportion large d'intrusions est réalisée via les réseaux et par conséquent l'utilisation des informations sur le trafic réseau rend l'IDS plus efficace. Ce problème motive l'évolution des IDS basés hôte vers l'IDS basé réseau. Le système de détection d'intrusions basé réseau détecte des attaques en capturant et analysant des paquets du réseau. Les avantages et les inconvénients [8,13] de ce type d'IDS sont :

7.2.1. *Avantages*

- L'IDS basé réseau est capable de contrôler un grand nombre d'hôte avec un petit coût de déploiement.
- Il n'influence pas sur les performances des entités surveillées.
- L'IDS basé réseau est capable d'identifier les attaques de multiples hôtes.
- L'IDS basé réseau assure une grande sécurité contre les attaques parce qu'il est invisible aux attaquants.

7.2.2. *Inconvénients*

- L'IDS basé réseau ne peut pas fonctionner dans des environnements cryptés.
- Ce type d'IDS ne permet pas d'assurer si une tentative d'attaque est couronnée de succès.

8. Les architectures d'implémentation des IDS

L'architecture d'implémentation d'un système de détection d'intrusions qui est considérée comme une stratégie de contrôle décrit la manière de contrôle effectuée par les éléments d'un système de détection d'intrusions. Nous distinguons trois approches d'implémentation [8, 24, 25,4] : ***Monolithique, hiérarchique*** et ***coopérative***.

8.1. L'approche monolithique (centralisée)

Les premières mises en œuvre des systèmes de détection d'intrusions ont employé une architecture monolithique sous laquelle les données rassemblées seront analysées à un point central. Puisque le contrôle de l'activité des utilisateurs d'un seul hôte ne révèle pas les attaques impliquant des hôtes multiples. L'IDS basé réseau a été développé, qui analysant le trafic de réseau pour déduire les anomalies venant du réseau.

Bien qu'un IDS basé réseau avec un serveur central a montré des résultats prometteurs pour des réseaux à petite échelle. Cette approche ne peut supporter un grand réseau à cause de la quantité énorme de données des différents hôtes qui doivent être analysée par le serveur central, ce qui engendre une dégradation sévère des performances du réseau. Un exemple d'un système de détection d'intrusions qui se base sur l'approche monolithique est le système NADIR [14, 26,27].

8.2. L'approche hiérarchique

Cette approche a été proposée pour surmonter les problèmes de l'approche monolithique. Elle est caractérisée par l'existence des secteurs de contrôle hiérarchiques. Chaque IDS contrôle un secteur avec l'élimination du transfert des données d'audit rassemblées par les hôtes locaux à un point central. Chaque IDS à n'importe quel niveau de contrôle exécute une analyse locale et envoie ses résultats d'analyse au niveau suivant dans la hiérarchie. L'approche hiérarchique montre la meilleure incrémentabilité « scalability » en permettant des analyses locales aux secteurs de contrôle distribués. Cependant, les problèmes vus précédemment demeurent toujours. En plus, le changement de la topologie du réseau cause un changement aussi bien dans la hiérarchie de réseau est que dans les mécanismes de rassemblement des rapports d'analyse locaux. Ainsi, la difficulté de détecter les attaques qui visent le niveau le plus haut de la hiérarchie.

Un exemple de système de détection d'intrusions hiérarchique : GrIDS [28,29], EMERALD [30].

9.2. NIDES

NIDES (Next- Generation IDES) [14,26] est une version améliorée du système de détection d'intrusions IDES. Il assure la détection d'intrusions sur plusieurs hôtes (distribuées) en se basant toujours sur les données d'audit. Il n'y a aucune analyse du trafic réseau. Il utilise les mêmes algorithmes qu'IDES.

9.3. NADIR

NADIR (Network Anomaly Detection and Intrusion Reporter) [14,26,27] est un système expert qui a été conçu pour le réseau ICN (Integrated Computing Network) du Laboratoire National Los Alamos. Son but est d'analyser les activités réseaux des utilisateurs et d'ICN en se basant sur les règles du système expert qui définissent la politique de sécurité et les comportements suspects. L'inconvénient majeur de ce système est qu'il ne peut être porté sur d'autres réseaux, étant donné que les protocoles réseaux d'ICN ne sont pas standards.

9.4. DIDS

DIDS (Distributed Intrusion Detection System) [14,26,27]est un système de détection d'intrusions basé réseau qui se base sur l'approche hiérarchique. Afin d'éviter la dégradation des performances de système, DIDS délègue certaines analyses locales aux hôtes locaux. Son architecture se compose de trois entités:

- Le « Host Monitor » : Il en existe un par hôte. Il collecte les données de l'hôte

  surveillé, fait une première analyse simple sur ces données puis transmet les événements pertinents au « DIDS Director ».
- Le « LAN Monitor » : Il en existe un pour chaque segment LAN. Il surveille le trafic

  sur le LAN, collecte les informations réseaux et reporte au « DIDS Director » les activités suspectes et non autorisées qui se sont produites sur le réseau.
- Le « DIDS Director » : Il analyse les rapports reçus du « LAN Monitor » et

des « Host

Monitor » afin de détecter les attaques potentielles.

9.5. GrIDS

GrIDS (Graph-Based Intrusion Detection System) [28, 29] a été conçu pour détecter des attaques à grande échelle. GrIDS considère les réseaux larges comme une agrégation de sous réseaux. Les données concernant l'activité des hôtes et le trafic réseau entre ces hôtes sont rassemblées dans des graphes d'activité qui révèlent la structure causale de l'activité réseau. Les nœuds d'un graphe d'activité correspondent aux hôtes constituant le réseau alors que les arêtes représentent l'activité réseau entre les différents hôtes.

Durant la phase de détection, GrIDS analyse les caractéristiques des graphes d'activité et compare ces graphes à des formes intrusives connues. S'il y a des similitudes entre ces graphes et des attaques connues, il en informe l'officier de sécurité.

9.6. CSM

CSM (Cooperating Security Manager) [26,27] est un système de détection d'intrusions qui peut être utilisé dans un environnement de réseau distribué. Son principal objectif est de détecter les activités intrusives de façon non centralisée car utiliser un directeur central qui coordonnerait toutes les activités limiterait la taille du réseau « le problème d'incrémentabilité ». Pour cela, CSM doit s'exécuter sur chaque hôte connecté au réseau. Ainsi, au lieu de reporter les activités anormales à un directeur central, les CSM communiquent entre eux pour détecter d'une manière coopérative les intrusions réseaux. Les composants principaux de ce système de détection d'intrusions sont :

- Un système de détection d'intrusions local (IDS) : qui assure la détection d'intrusions pour un hôte local.
- Un gestionnaire de sécurité : qui coordonne la détection d'intrusions distribuée entre les CSM.

- Un gestionnaire d'intrus (IH : intruder handling component) : dont le rôle est d'entreprendre les actions nécessaires lorsqu'une intrusion est détectée.

9.7. AAFID

Le système AAFID (Autonomous Agent for Intrusion Detection) [32, 33,31] est la première tentative d'utilisation des agents autonomes pour les systèmes de détection d'intrusions basés réseau où plusieurs agents indépendants opèrent de manière coopérative pour assurer la surveillance du système cible. La décision finale du système est le résultat de coopération entre ces différents processus.

10. Conclusion

Ces dernières années, les systèmes de détection d'intrusions ont gagné une place importante dans la conception de la sécurité des systèmes d'information. Ils sont largement déployés dans les entreprises pour diverses raisons telles que : la documentation des attaques, l'évaluation de la sécurité, et plus généralement la surveillance des systèmes d'information pour arrêter les attaques afin de limiter les dégâts.

Les systèmes de détection d'intrusions se caractérisent principalement par :

- La méthode de détection, on distingue deux principales méthodes : la détection par signatures et la détection d'anomalies. Ces deux méthodes présentent des avantages et des inconvénients,
- Les sources d'information, qui peuvent être : le réseau, l'hôte et les applications,
- Le comportement de la détection, qui peut être passif ou actif,
- La synchronisation entre les sources et le détecteur, qui peut être périodique ou continue pour les IDS temps réel.

Les premiers systèmes de détection d'intrusions fonctionnaient indépendamment les uns des autres sans problèmes, parce que les environnements étaient relativement simples. Actuellement, les systèmes sont devenus plus complexes avec une très forte connectivité, que se soit localement dans des grands réseaux locaux ou

métropolitains, ou vers des réseaux plus grands tels que l'Internet. Cet environnement plus complexe a créé très naturellement le besoin de communication et d'interopérabilité.

Dans les prochains chapitres nous allons voir plus en détails les efforts faits pour offrir un standard d'interopérabilité et de coopération entre les IDS.

Enfin, notons que les systèmes de détection d'intrusions ne sont pas la solution miracle, ils possèdent beaucoup de limites et problèmes liés à plusieurs facteurs tels que : l'observation des événements, le facteur humain, et les attaques par déni de service.

Cependant, ils représentent des outils très efficaces s'ils sont déployés dans une bonne architecture de la sécurité impliquant plusieurs outils.

Chapitre II :

LA LOGIQUE DE DESCRIPTION

d'individus représenté par D. Une base de connaissances se compose alors d'une hiérarchie de concepts et d'une (éventuelle) hiérarchie de rôles.

Les opérations qui sont à la base du raisonnement terminologique sont la classification et l'instanciation. La classification s'applique aux concepts, le cas échéant aux rôles, et permet de déterminer la position d'un concept et d'un rôle dans leurs hiérarchies respectives ; la construction et l'évolution de ces hiérarchies est ainsi assistée par le processus de classification. L'instanciation permet de retrouver les concepts dont un individu est susceptible d'être une instance.

2. Un historique des logiques de description

Le développement des LD fut fortement influencé par les travaux sur la logique des prédicats, les schémas (frames) (Minsky, 1981) et les réseaux sémantiques. Des correspondances existent entre les LD et ces formalismes [34,35]. La présence de catégories générales d'objets et de relations fait d'ailleurs partie de l'héritage conceptuel des schémas et des réseaux sémantiques.

2.1. La première génération de logiques de description (1980 - 1990)

Les premiers travaux sur les LD commencèrent au début des années 1980 avec des systèmes à base de connaissances tels que KL-ONE, BACK et LOOM [36,37]. Ces premières implantations résolvent des problèmes d'inférence en temps souvent polynomial, par le biais d'une catégorie d'algorithmes de vérification de subsomption de type normalisation/comparaison (structural subsumption algorithmes). Ces algorithmes ne s'appliquent qu'à des LD peu expressives, sans quoi ils sont incomplets, c'est-à-dire qu'ils sont incapables de prouver certaines formules vraies.

2.2. La deuxième génération de logiques de description (1990)

Dans les années 1990, une nouvelle classe d'algorithmes est apparue : les algorithmes de vérification de satisfiabilité à base de tableaux (tableau-based algoritdms). Ces derniers raisonnent sur des LD dites expressives ou très expressives, mais en temps exponentiel. Cependant, en pratique, le comportement des algorithmes est souvent

acceptable [36]. L'expressivité accrue a ouvert la porte à de nouvelles applications telles que le Web sémantique [36, 38,39]. Le terme logique de description expressive (LDE) désigne l'ensemble des LD qui ont émergé pendant cette période.

3. Les logiques de description

Les logiques de description [11] aussi appelées logiques descriptives (LDs) peuvent être utilisées pour représenter la connaissance terminologique d'un domaine d'application d'une manière formelle et structurée. Le nom de logique de description se rapporte, d'une part à la description de concepts utilisée pour décrire un domaine et d'autre part à la sémantique basée sur la logique qui peut être donnée par une transcription en logique des prédicats du premier ordre. La logique de description a été développée comme une extension des frames et des réseaux sémantiques, qui ne possédaient pas de sémantique formelle basée sur la logique.

4. Les deux niveaux de description

La modélisation des connaissances d'un domaine avec les LD se réalise en deux niveaux. Le premier, le niveau terminologique ou TBox, décrit les connaissances générales d'un domaine alors que le second, le niveau factuel ou ABox, représente une configuration précise. Une TBox comprend la définition des concepts et des rôles, alors qu'une ABox décrit les individus en les nommant et en spécifiant en termes de concepts et de rôles, des assertions qui portent sur ces individus nommés. Plusieurs ABox peuvent être associés à une même TBox ; chacune représente une configuration constituée d'individus, et utilise les concepts et rôles de la TBox pour l'exprimer.

4.1. Le niveau terminologique (TBox) :

Les entités atomiques

Les concepts atomiques et rôles atomiques constituent les entités élémentaires d'une TBox.

Les concepts et rôles atomiques prédéfinis

Les LD prédéfinissent minimalement quatre concepts atomiques : le concept $\top$ et le rôle $\top_R$, les plus généraux de leur catégorie respective, et le concept $\bot$ ainsi que le rôle $\bot_R$ les plus spécifiques (c'est-à-dire l'ensemble vide).

Les entités composées

Les concepts et rôles atomiques peuvent être combinés au moyen de constructeurs pour former respectivement des concepts et des rôles composés. Les différentes LD se distinguent par les constructeurs qu'elles proposent.

Le constructeur and ($\cap$) permet de définir une conjonction d'expressions conceptuelles. Le constructeur not ($\neg$) correspond à la négation et ne porte que sur les concepts primitifs.

La quantification universelle all ($\forall$ r.C) précise le co-domaine du rôle r; la quantification existentielle non typée somme ($\exists$ r) introduit le rôle r et affirme l'existence d'(au moins) un couple d'individus en relation par l'intermédiaire de r.

4.2. Le niveau factuel (ABox)

Une ABox contient un ensemble d'assertions sur les individus : (1) des assertions d'appartenance et (2) des assertions de rôle. Chaque ABox doit être associé à une TBox, car les assertions s'expriment en termes de concepts et des rôles de la TBox. Voire la figure 2.1 et le tableau 2.1.

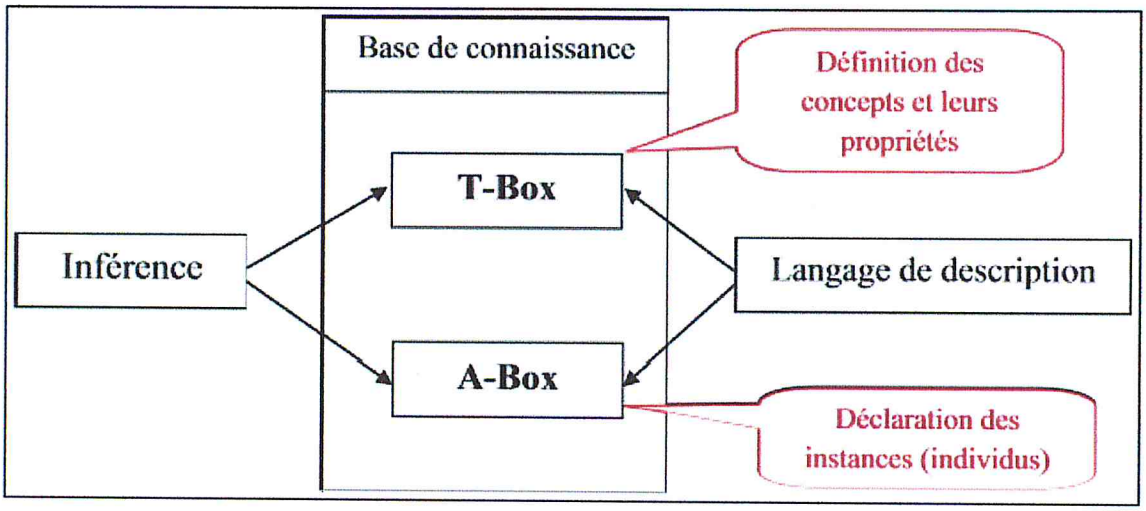


Figure 2.1 : Le Système a base de la logique de description

Un exemple sur la TBOX et L'ABOX

TBox	*ABox*
$Femelle \sqsubseteq \top \sqcap \neg Mâle$	*Humain(djamila)*
$Mâle \sqsubseteq \top \sqcap \neg Femelle$	*Femelle(djamila)*
$Animal \equiv Mâle \sqcup Femelle$	*Femelle(ryme)*
$Humain \sqsubseteq Animal$	*Humain(wahib)*
$Femme \equiv Humain \sqcap Femelle$	*¬ Femelle(wahib)*
$Homme \equiv Humain \sqcap \neg Femelle$	*Homme(mustapha)*
$Mère \equiv Femme \sqcap \exists relationParentEnfant$	*relationParentEnfant(djamila,ryme)*
$Père \equiv Homme \sqcap \exists relationParentEnfant$	*relationParentEnfant(wahib,mustapha)*
$MèreSansFille \equiv Mère \sqcap$ $\forall relationParentEnfant.\neg Femme$	
$relationParentEnfant \sqsubseteq \top_R$	

Tableau 2.1 : Une base de connaissances composée d'une TBox et d'une ABox

5. La logique minimale d' $\mathcal{AL}$

Pour des fins de simplicité, ce document décrit en premier lieu une logique minimale nommée $\mathcal{AL}$ qui a été introduite par [60] et qui revêt d'une grande importance dans le domaine. Cette logique est minimale, dans le sens où une logique moins expressive représente peu d'intérêt [35].

5.1. Les constructeurs d' $\mathcal{AL}$

La figure 2.2 illustre les constructeurs offerts par $\mathcal{AL}$ pour l'édification de concepts composés.

$C, D \longrightarrow$	A	(concept atomique)
	$\mid \top$	(le concept universel)
	$\mid \bot$	(le concept le plus spécifique)
	$\mid \neg A$	(la négation atomique)
	$\mid C \sqcap D$	(l'intersection)
	$\mid \exists R.\top$	(quantification existentielle limitée)
	$\mid \forall R.C$	(quantification universelle complète)

Figure 2.2 La grammaire des expressions conceptuelles selon $\mathcal{AL}$

Le constructeur C ⋂ D permet de faire la conjonction de deux concepts composés, ce qui représente l'ensemble des individus, membres à la fois du concept C et du concept D pour une interprétation. Le constructeur ¬A est utilisé pour évoquer la négation d'un concept atomique, c'est-à-dire les individus pour une interprétation qui n'appartiennent pas au concept atomique A.

Le quantificateur existentiel non typé ∃R. ⊤ désigne l'ensemble des individus, membres du domaine d'un rôle R pour une interprétation donnée.

Le quantificateur universel ∀R.C évoque l'ensemble des individus du domaine d'un rôle R qui sont en relation, par le biais de R, avec un individu du concept C, pour une interprétation donnée. $\mathcal{AL}$ ne permet pas la spécification de rôles à l'aide de constructeurs (rôles composés).

5.2. Les extensions d' $\mathcal{AL}$

Il existe trois façons proéminentes d'étendre $\mathcal{AL}$: (1) ajouter des constructeurs de concepts, (2) ajouter des constructeurs de rôles et (3) énoncer des contraintes sur l'interprétation des rôles [40].

L'extension par ajout de constructeurs de concepts ou de rôles

Le tableau 2.2 illustre des exemples de constructeurs pour augmenter $\mathcal{AL}$ [40]. La première colonne contient la lettre qui désigne le constructeur, la deuxième sa syntaxe d'utilisation et la dernière sa sémantique. La nomenclature des LD dicte que pour chaque constructeur ajouté, il faut agglutiner la lettre correspondante au nom de la logique originale.

$\lvert\mathcal{O}\rvert$	$\{a_1, a_2, ..., a_n\}$	$\{a_1^{\mathcal{I}}, a_2^{\mathcal{I}}, ..., a_n^{\mathcal{I}}\}$
$\lvert\mathcal{U}\rvert$	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\lvert\mathcal{E}\rvert$	$\exists R.C$	$\{a \in \triangle^{\mathcal{I}} \lvert \{\exists b.(a,b) \in R^{\mathcal{I}}\} \wedge b \in C^{\mathcal{I}}\}$
$\lvert\mathcal{C}\rvert$	$\neg C$	$\triangle^{\mathcal{I}} \backslash C^{\mathcal{I}}$
$\lvert\mathcal{I}\rvert$	R_1^{-1}	$\{(y,x) \lvert (x,y) \in R_1^{\mathcal{I}}\}$
$\lvert\mathcal{H}\rvert$	$R_1 \sqsubseteq R_2$	$R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$
$\lvert\mathcal{F}\rvert$	$= 1R$	$\{x \in \triangle^{\mathcal{I}} \lvert\lvert \{y \in \triangle^{\mathcal{I}} \lvert (x,y) \in R^{\mathcal{I}}\}\rvert = 1\}$
	$\geq 2R$	$\{x \in \triangle^{\mathcal{I}} \lvert\lvert \{y \in \triangle^{\mathcal{I}} \lvert (x,y) \in R^{\mathcal{I}}\}\rvert \geq 2\}$
$\lvert\mathcal{N}\rvert$	$\geq nR$	$\{a,b \in \triangle^{\mathcal{I}} \lvert\lvert (a,b) \in R^{\mathcal{I}}\}\rvert \geq n\}$
	$\leq nR$	$\{a,b \in \triangle^{\mathcal{I}} \lvert\lvert (a,b) \in R^{\mathcal{I}}\}\rvert \leq n\}$
	$= nR$	$\{a,b \in \triangle^{\mathcal{I}} \lvert\lvert (a,b) \in R^{\mathcal{I}}\}\rvert = n\}$
$\lvert\mathcal{Q}\rvert$	$\geq nR.C$	$\{a,b \in \triangle^{\mathcal{I}} \lvert\lvert (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\rvert \geq n\}$
	$\leq nR.C$	$\{a,b \in \triangle^{\mathcal{I}} \lvert\lvert (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\rvert \leq n\}$
	$= nR.C$	$\{a,b \in \triangle^{\mathcal{I}} \lvert\lvert (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\rvert = n\}$

Tableau 2.2 Exemple de constructeurs de rôles et concepts pour étendre $\mathcal{AL}$

L'extension par ajout de contraintes sur l'interprétation des rôles

La spécification d'un ensemble de rôles transitifs $\mathcal{N}_{\mathcal{R}+}$, constitue $_{\mathcal{R}+}$ une extension par ajout de contraintes sur l'interprétation des rôles (désignée par la lettre $_{\mathcal{R}+}$), qui permet l'expression de rôles transitifs tels qu'*ancêtreDe* ou *frèreDe* [40]. La lettre $\mathcal{S}$ désigne la logique $\mathcal{ALC}$ additionnée de $_{\mathcal{R}+}$.

L'extension des types primitifs ($\mathcal{D}$)

Une dernière extension, symbolisée par la lettre ($\mathcal{D}$), ajoute le support des types primitifs [39]. Cette extension augmente $\mathcal{AL}$ d'un second domaine d'interprétation $\Delta_D^{\mathcal{I}}$ disjoint avec $\Delta^{\mathcal{I}}$ et qui représente l'ensemble des valeurs de type primitif. Le domaine $\Delta_D^{\mathcal{I}}$ définit plusieurs sous-domaines tels que les entiers, les chaînes de caractères, les entiers positifs, etc. Les éléments de ces domaines se nomment individus primitifs. De plus, l'extension ajoute un nouveau type de rôle, défini comme une relation binaire sur $\Delta^{\mathcal{I}} \times \Delta_D^{\mathcal{I}}$ et appelé rôles à valeurs primitives. La lettre *U* représente l'ensemble de ces rôles. Cet ajout autorise la spécification d'assertions de rôle telles que *u(a, 205006007)* et *v(a, "Jean Jacques")* où u, v $\in$ U.

Quoiqu'il soit possible de définir un constructeur de hiérarchies pour ces rôles (par exemple : $u \sqsubseteq v$ avec la sémantique $u^{\mathcal{I}} \subseteq v^{\mathcal{I}}$), plusieurs constructeurs pour rôle classique tels que le constructeur de rôle transitif et le constructeur de rôle inverse n'ont pas leur équivalent pour les rôles à valeurs primitives.

6. L'INFERENCE

L'inférence s'effectue au niveau terminologique ou factuel. Les sections 6.1 et 6.2 abordent le raisonnement au niveau terminologique et factuel, respectivement. Pour terminer, on présente un tableau comparatif des différents moteurs d'inférence.

6.1. L'inférence au niveau terminologique

Quatre principaux problèmes d'inférence se présentent au niveau terminologique [35] :

- **Satisfiabilité** : Un concept C d'une terminologie $\mathcal{T}$ est satisfiable si et seulement existe un modèle $\mathcal{I}$ de $\mathcal{T}$ tel que $C^{\mathcal{I}} \neq \emptyset$.
- **Subsomption** : Un concept C est subsumé par un concept D pour une terminologie si et seulement si $C^{\mathcal{I}} \sqsubseteq D^{\mathcal{I}}$ pour tout modèle $\mathcal{I}$ de $\mathcal{T}$.
- **Équivalence** : Un concept C est équivalent à un concept D pour une terminologie et seulement si $C^{\mathcal{I}} \equiv D^{\mathcal{I}}$ pour chaque modèle $\mathcal{I}$ de $\mathcal{T}$.

- **Disjonction (disjointness)** : Des concepts C et D sont disjoints par rapport terminologie
  $\mathcal{T}$ si et seulement si $C^{\mathcal{I}} \setminus D^{\mathcal{I}} =$; pour chaque modèle $\mathcal{I}$ de $\mathcal{T}$.

Les moteurs d'inférence actuels tirent généralement profit du fait que les quatre types de problèmes d'inférence peuvent être réduits à des problèmes de subsomption ou à des problèmes de satisfiabilité. Les figures 2.3 et 2.4 illustrent cette propriété qui implique, que les moteurs d'inférence des LD ne nécessitent souvent qu'un seul algorithme d'inférence pour raisonner au niveau terminologique [35] . D'ailleurs, les deux grandes classes d'algorithmes de raisonnement pour les LD (algorithmes de subsomption de type normalisation/comparaison et algorithmes de vérification de satisfiabilité à base de tableaux) correspondent aux façons de réduire respectivement des problèmes d'inférence à des problèmes de subsomption et de satisfiabilité [36].

C est insatisfiable	$\Longleftrightarrow$	C est subsumé par $\bot$
C et D sont équivalents	$\Longleftrightarrow$	C est subsumé par D, et D par C
C et D sont disjoints	$\Longleftrightarrow$	$C \sqcap D$ est subsumé par $\bot$

Figure 2.3 Réduction des problèmes d'inférence d'une TBox à des problèmes de subsomption

C est subsumé par D	$\Longleftrightarrow$	$C \sqcap \neg D$ est insatisfiable
C et D sont équivalents	$\Longleftrightarrow$	$C \sqcap \neg D$ et $\neg C \sqcap D$ sont insatisfiables
C et D sont disjoints	$\Longleftrightarrow$	$C \sqcap D$ est insatisfiable

Figure 2.4 Réduction des problèmes d'inférence d'une TBox à des problèmes de satisfiabilité

Complexité	Logiques de description
P	$\mathcal{AL}$, $\mathcal{ALN}$
NP	$\mathcal{ALE}$
PSpace	$\mathcal{ALC}$, $\mathcal{ALEN}$
ExpTime	$\mathcal{SHIQ}$, $\mathcal{SHOQ}$
NExpTime	...

Tableau 2.3 Complexité de la vérification de la subsomption et de la satisfiabilité en fonction de l'expressivité des LD

Le tableau 2.3 présente un aperçu non exhaustif de la complexité du raisonnement au niveau terminologique en fonction de l'expressivité [42, 36, 41]. Ce tableau met en évidence la connaissance pointue des LD que la communauté scientifique détient. Les

classes de complexité énumérées dans le tableau proviennent de la théorie de la complexité informatique. Voici une définition de ces classes [43] :

- **P**: la classe des problèmes de décision (un problème de décision prend en entrée un énoncé de problème et produit en sortie une réponse positive ou négative: oui ou non). qui requièrent un temps polynomial par rapport à la taille du problème pour obtenir une solution avec une machine de Turing déterministe.
- **NP**: la classe des problèmes qui nécessitent un temps polynomial pour trouver une solution avec une machine de Turing non déterministe.
- **PSpace** : la classe des problèmes de décision qui requièrent une quantité de mémoire polynomiale pour résoudre un problème avec une machine de Turing déterministe ou non déterministe.
- **ExpTime** : la classe des problèmes de décision solvables par une machine de Turing déterministe en un temps $\Theta(2^{p(n)})$ où $p(n)$ est une fonction polynomiale de n, la taille du problème.
- **NExpTime** : la classe des problèmes de décision solvables par une machine de Turing non-déterministe en un temps $\Theta(2^{p(n)})$ où $p(n)$ est une fonction polynomiale de n, la taille du problème.

Il est connu que P $\subseteq$ NP $\subseteq$ PSpace $\subseteq$ ExpTime $\subseteq$ NExpTime [43].

6.2. L'inférence au niveau factuel

Le niveau factuel comprend quatre principaux problèmes d'inférence [35] : **Cohérence** : Une ABox A est cohérente par rapport à une TBox $\mathcal{T}$ si et seulement s'il existe un modèle $\mathcal{I}$ de A et $\mathcal{T}$.

- **Vérification d'instance** : Vérifier par inférence si une assertion C(a) est vraie pour tout modèle $\mathcal{I}$ d'une ABox A et d'une TBox $\mathcal{T}$.
- **Vérification de rôle** : Vérifier par inférence si une assertion R(a, b) est vraie pour tout modèle $\mathcal{I}$ d'une ABox A et d'une TBox $\mathcal{T}$.
- **Problème de récupération** : Pour une ABox A, un concept C d'une terminologie $\mathcal{T}$, inférer les individus $a^{\mathcal{I}}1$ ...$a^{\mathcal{I}}n \in$ CI pour tout modèle $\mathcal{I}$ de $\mathcal{T}$.

Les moteurs d'inférences

Le tableau 3.4 dresse une comparaison des principaux moteurs d'inférence pour les LDE: FaCT [44], Racer [45], Pellet [46], FaCT++ [38] , F-OWL [47] Surnia, et Hoolet. Le critère "Mise-à-échelle" mesure la capacité à demeurer efficace

proportionnellement à la complexification des ontologies. Le tableau reprend les données de [38] pour ce critère, celui de décidabilité et pour les caractéristiques de Hoolet, Surnia et F-OWL. Ce document ne traitera pas du critère "OWL" dans le tableau 1.4.

Moteur	*Racer*	*FaCT*	*Pellet*	*FaCT++*
LD	$\mathcal{SHIQ}(\mathcal{D})-$	$\mathcal{SHIQ}$, $\mathcal{SHF}$	$\mathcal{SHIN}(\mathcal{D})$, $\mathcal{SHON}(\mathcal{D})$	$\mathcal{SHIF}(\mathcal{D})$
Implantation	*C++*	*Common Lisp*	*Java*	*C++*
Inférence	*TBox/ABox*	*TBox*	*TBox/ABox*	*TBox*
API *Java*	oui	oui	natif	oui
Mise-à-échelle	bonne	bonne	bonne	bonne
OWL	*OWL-DL*$\sim^{\dagger}$	*OWL-DL*$\sim^{\dagger}$	*OWL-DL*$\sim^{\dagger}$	*OWL-LITE*
Décidabilité	oui (*OWL-LITE*)	oui	oui (*OWL-LITE*)	oui
DIG	oui	oui	non	?
Moteur	*Surnia*	*Hoolet*	*F-OWL*	
LD	logique prédicats	logique prédicats	$\mathcal{SHIQ}(\mathcal{D})$ et *RDF*	
Implantation	*Python*	*Java*	*Java*	
Inférence	*TBox/ABox*	*TBox/ABox*	*TBox/ABox*	
API *Java*	?	oui	oui	
Mise-à-échelle	médiocre	médiocre	médiocre	
OWL	*OWL-FULL*$\sim^{\dagger}$	*OWL-DL*$\sim^{\dagger}$	*OWL-FULL*$\sim^{\dagger}$	
Décidabilité	non	non	non	
DIG	non	non	non	

Tableau 3.4 Tableau comparatif des principaux moteurs d'inférence pour LD

Le raisonnement sur TBox ou ABox

Tous ces moteurs raisonnent autant sur des ABox que sur des TBox, mis à part FaCT et FaCT++ qui se spécialisent en raisonnant sur des TBox seulement. FaCT++ est une variante de FaCT implantée en C++ pour une efficacité accrue, et avec quelques ajouts et différences tels que le tableau l'illustre.

L'expressivité et l'efficacité des moteurs d'inférence

Les moteurs F-OWL, Hoolet et Surnia raisonnent sur des logiques de description très expressives. Hoolet et Surnia raisonnent sur l'expressivité totale de la logique des prédicats, alors que F-OWL infère sur la logique $\mathcal{SHIQ}(\mathcal{D})$ et sur l'expressivité totale

de RDF (Ressource Description Framework, un modèle RDF représente un domaine par un ensemble de triplets sujet, prédicat, valeur qui lient par des propriétés (prédicat) des ressources entre elles (sujet, valeur)).Ces trois moteurs se basent sur des méthodes expérimentales de raisonnement qui exhibent des performances intéressantes pour des problèmes simples, mais insuffisantes pour une utilisation industrielle, en raison de leur faible efficacité et de la non-décidabilité de leurs algorithmes. Ce texte les mentionne à titre informatif.

Comme l'indique le critère "Mise-à-échelle" dans le tableau, les moteurs les plus performants actuellement sont Racer, FaCT, FaCT++ et Pellet. FaCT, FaCT++ et Racer disposent probablement de la plus grande notoriété actuellement, chacun d'eux utilisé dans de nombreux projets. Le moteur Pellet est beaucoup plus récent que les autres et encore en développement. Il échoue à atteindre le niveau de performance de FaCT et Racer [46].

7. CONCLUSION

Dans ce chapitre, nous avons présenté les logiques de descriptions, qui constituent un formalisme de représentation de connaissances caractérisé par les points suivants :

- Un langage qui permet de construire des descriptions conceptuelles génériques (concepts primitifs et définis) ou individuelles (instances).
- Une sémantique associée à chaque construction syntaxique par l'intermédiaire d'une interprétation.
- Une relation de subsomption qui permet d'organiser les descriptions par niveau de généralité, et de procéder à des inférences ; cette relation est à la base des processus de classification et d'instanciation.

Sur le plan de la théorie de la représentation des connaissances, les logiques de descriptions ont apporté une certaine originalité dans le traitement de problèmes d'intelligence artificielle. Elles se sont avérées être un des premiers formalismes

structurels où sont manipulées des structures permettant d'étudier explicitement le niveau syntaxique et le niveau sémantique dans une représentation, ainsi que les relations existant entre ces deux niveaux, comme c'est le cas en logique classique.

Chapitre III :

L'INTEROPERABILITE ENTRE LES SYSTEMES DE DETECTION D'INTRUSIONS

1. Introduction

Ces dernières années, l'utilisation intensive des systèmes de détection d'intrusions a mis en évidence le problème de la gestion d'un flux important d'alertes générées par ces systèmes. De plus, la majorité des systèmes de détection d'intrusions utilisent leur propre format d'alerte, ce qui a rendu difficile la construction d'une image globale à partir de l'analyse des alertes provenant de plusieurs détecteurs hétérogènes.

Pour toutes ces raisons, le problème des systèmes de détection d'intrusions est : « d'être capable d'analyser et de réagir devant un énorme volume d'alertes, générées dans des formats différents, et avec un taux de fausses alertes très élevé ».

Devant un environnement de détection d'intrusions caractérisé par un taux de détection très faible, un taux très élevé de fausses alertes, et une granularité de l'information contenue dans les alertes très faible et qui diffère d'un détecteur à un autre,un énorme effort a été fourni par la communauté de la détection d'intrusions pour la standardisation des formats d'alertes générées par les systèmes de détection d'intrusions, ce qui a permis d'offrir un espace de communication plus ouvert entre les outils de sécurité.

Dans ce chapitre, nous présentons les efforts de standardisation faits dans le domaine de l'interopérabilité entre plusieurs systèmes de détection d'intrusions.

2. Interopérabilité entres plusieurs systèmes de détection d'intrusions

Dans les grands réseaux, plusieurs systèmes de détection d'intrusions et d'autres activités d'audit de plusieurs propriétaires existent et continuent à exister. Un modèle de données standard pour la représentation des événements et des attaques se produisant dans ces réseaux s'avère nécessaire pour la corrélation des alertes générées par des sources hétérogènes.

En 1997, DARPA a initié le projet de recherche CIDF (Common Intrusion Detection

Framework) dans le but de coordonner les différents projets financés par DARPA et assurer l'interopérabilité entre les outils qui en résultent [48].

Les développeurs de ce projet ont mis en place un modèle permettant l'interopérabilité entre les différents composants d'un système de détection d'intrusions. Cet effort a été complété par le langage CISL (Common Intrusion Specification Language) qui assure la représentation et la communication des données entre ces composants.Malgré la puissance du langage CISL, ce dernier n'a pas été adopté par la plupart des industriels, ce qui a mis fin au projet en 1999.

L'échec du CIDF a motivé la création du groupe de travail IDWG dans l'IETF, avec la participation de plusieurs chercheurs du CIDF. L'objectif de ce groupe est la définition d'un format de données et des procédures pour le partage des informations entre : les systèmes de détection d'intrusions, les systèmes de réponses (CERT : Center Engineering Response Team), et les consoles de gestion. Contrairement au CIDF qui restait un simple projet de recherche, l'IDWG a essayé de proposer des standards pour l'interopérabilité entre les systèmes de détection d'intrusions. Une des ses propositions est le modèle IDMEF (Intrusion DetectionMessage Exchange Format) [49], qui est un modèle de données utilisé pour reporter les alertes. Ce modèle est conçu en orienté objet et implémenté en XML, ce qui a facilité son adoption à la fois par les industriels et le monde académique.

Une autre proposition, est le protocole IDXP (Intrusion Detection eXchange Protocol), qui est un protocole de niveau application développé pour l'échange sécurisé des messages IDMEF. Il est implémenté en partie comme un BEEP (Blocks Extensible

Exchange Protocol). Ce protocole permet l'utilisation de l'authentification, la confidentialité et le Tunnel pour le Firewall.

Dès le début des années 2000, plusieurs études sur les systèmes de détection d'intrusions ont confirmé que l'un des grands problèmes des systèmes de détection d'intrusions est le taux très élevé de fausses alertes [48]. Les fausses alertes consomment du temps d'analyse, et elles peuvent faire dévier accidentellement les intentions sur les vraies attaques. Tout simplement, un nombre très élevé de fausses alertes peut gêner les opérateurs de sécurité au point d'ignorer tous les messages d'avertissements.

Pour améliorer le taux de détection et diminuer l'impact des fausses alertes, il est nécessaire de faire coopérer un ensemble de SDI. Plus précisément, il s'agit de raisonner sur les alertes générées par un ensemble de SDI distribuées dans le système d'information surveillé. Un Système de Détection d'Intrusions Coopérative (SDIC) est composé d'un ensemble de SDI et d'une unité de corrélation d'alertes. Un SDIC a l'avantage de détecter les attaques survenant sur tout le système surveillé en corrélant les événements provenant de différents composants (application, hôte, sous réseaux, etc.). Un tel système a également l'avantage de réduire les coûts en partageant les ressources entre les sondes.

2.1. Modèle CIDF

Le modèle CIDF [50] a été établi dans le but de préciser les différents composants formant un SDI classique, ainsi que les interactions qui apparaissent entre ces composants. Nous observons ainsi quatre composants (voir la Figure 3.1) :

- E-Box (générateur d'événements) : son but est de fournir des événements aux autres composants. Un événement peut être représenté par un paquet particulier,

par une séquence de plusieurs trames, ou carrément par un flux continu de données (pour représenter par exemple une connexion TCP). Une information temporelle (la date et l'heure de capture de ces trames) est souvent associée à ces événements. Un E-Box représente donc le point de contact entre le SDI-R et le réseau qu'il est censé surveiller. En pratique, un E-Box pourrait par exemple être un sniffer classique. À l'origine, d'ailleurs, les premiers SDI-R étaient uniquement composés de sniffer. Mentionnons que un E-Box pourra en outre déjà réaliser certains filtrages via des filtres tels que BPF (Berkley Packet Filter), pour n'observer par exemple qu'une partie du trafic a destination d'un serveur donné, ou encor destiné a une portion précise du réseau. En outre, un SDI-R complet peut bien évidemment comporter plusieurs E-BOX, qui seront alors placés a des endroits stratégiques du réseau.

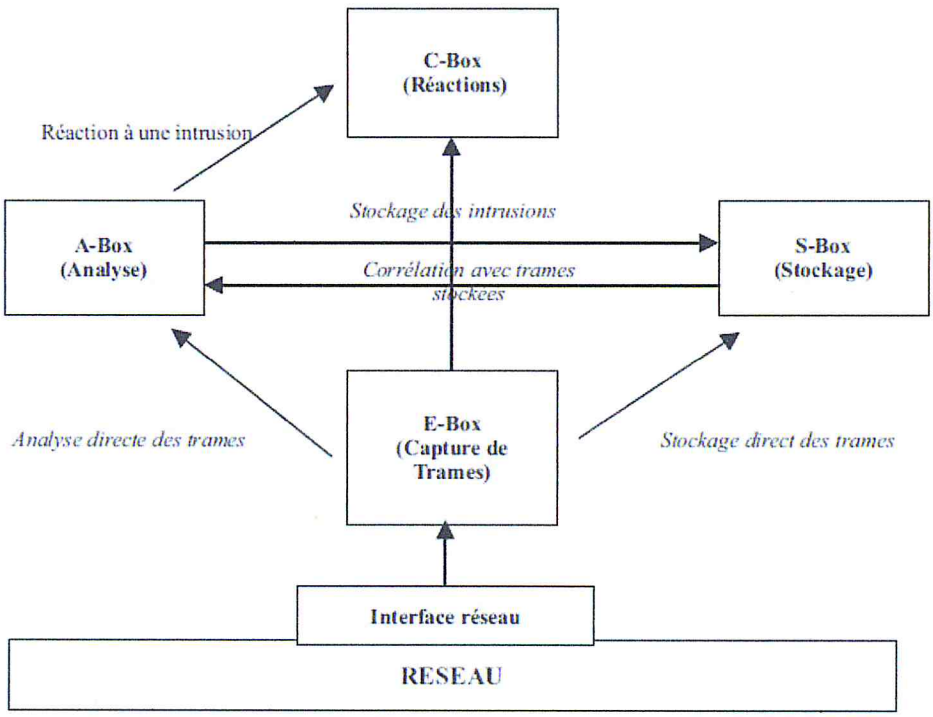


Figure 3.1 Modèle CIDF

- S-BOX (mécanisme de stockage) : une fois les données capturées, une analyse brutale en temps réel n'est probablement pas la meilleure solution. Ces données peuvent êtres archivées, dans un format précis, via un dispositive de stockage. Cet emplacement de stockage peut aussi bien êtres un fichier contenant les trames brutes capturées du réseau, qu'une base de données SQL dans laquelle les champs de ces trames seront encodés. De nouveau, plusieurs S-BOX peuvent êtres utilisés en parallèle, de façon complémentaires (distribution des données) ou redondantes (pour en assurer la sécurité).
- A-BOX (moteur d'analyse) : les données étant disponibles sous forme brute, il est maintenant temps d'analyser celle-ci, de manière à rechercher effectivement des tentatives d'intrusions. C'est le rôle précis d'un A-BOX. Notons principalement qu'un A-BOX peut travailler sur les données fournies par les E-BOX (analyse en temps réel), sur les données fournies par les S-BOX (analyse différée), ou encore sur les données déjà mémorisées. Enfin, les résultats de cette analyse, peuvent à leur tour êtres journalisées dans un S-BOX, ou simplement affichés sur une console, à disposition de l'analyste.
- C-Box (contre-mesures) : une fois l'intrusion détectée, des réactions peuvent être envisagées. Celles-ci peuvent aller de la fermeture d'une connexion TCP en générant des paquets de réponse nommés rst, à la modification de règles de filtrage directement sur un pare-feu/ routeur. Notons qu'un C-Box n'est pas nécessaire au fonctionnement correct d'un SDI-R, et dans certains cas extrêmes, peut même nuire à son fonctionnement. C'est pourquoi un C-Box ne fonctionne pas toujours de manière automatisée (l'accord d'une personne est exigé avant l'application de toute mesure défensive).

De plus, le CIDF définit les moyens de communication entre les différents composants(Boxes), incluant CISL (Common Intrusion Detection Language), GIDO (Generalized Intrusion Detection Objects), et une interface de programmation (AP I). En théorie, ceci signifie que les E-Boxes des vendeurs A, B, et C doivent être capables d'émettre des événements à une A-Box d'un

vendeur D. Cependant, dans la pratique, le déploiement de différents SDIs ensemble est un problème très difficile à résoudre.

En plus du CIDF, il existe d'autres travaux de standardisation liés au domaine de la détection d'intrusions tels que : IDMEF, IDIP, SASL, et IDX P. Particulièrement, le modèle IDMEF (Intrusion Detection Message Exchange Format) sera présenté dans la section suivante comme un format standard, pour normaliser les messages d'alertes des SDIs.

2.2. Modèle IDMFE

Le standard IDMEF, résumé dans la Figure 3.2, est un modèle de données développé par l'IETF dans l'objectif de permettre aux différents outils participants à la détection d'intrusions de reporter les événements qu'ils jugent suspicieux sous un même format. Ce standard permet l'interopérabilité entre des systèmes : commerciaux, ouverts et de recherche, ce qui permettra aux utilisateurs de mieux déployer ces systèmes selon leurs points forts et faibles, pour optimiser leur implémentation. Le standard IDMEF a été développé pour résoudre principalement les problèmes suivants [49] :

- La granularité des informations contenues dans les alertes diffère d'un système à un autre. Certaines alertes (d'un SDI donné) contiennent très peu d'information tels que : la source, la destination et le temps, alors que d'autres offrent plus d'information tels que : le service, l'utilisateur, le processus, etc.
- Les environnements de la détection sont différents, certains détecteurs utilisent l'analyse de trafic réseau, d'autres utilisent les Logs système et l'audit des applications.
- Les capacités des détecteurs sont différentes, elles dépendent de leur environnement, les détecteurs peuvent donner peu d'information dans les alertes ou des alertes plus détaillées.

D'après les problèmes cités précédemment, le modèle de données doit être suffisamment flexible pour supporter les différences d'information des différents types d'alertes. Pour cette raison le modèle est conçu par une approche orienté objet et implémenté en XML.

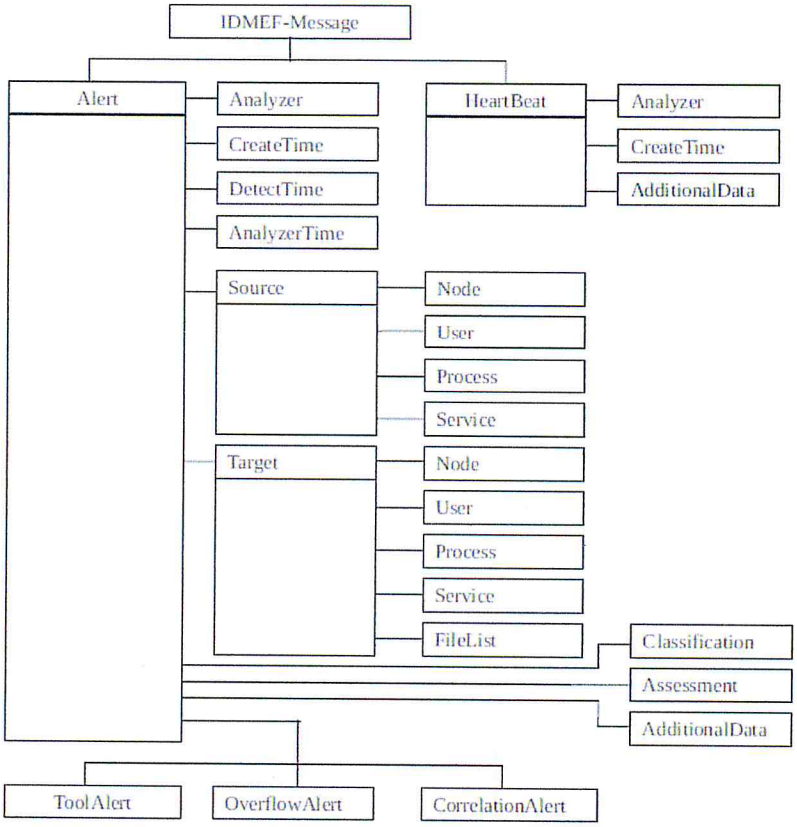


Figure 3.2 Modèle IDMFE

Déploiement de l'IDMEF

Le meilleur déploiement du modèle IDMEF est sous forme d'une ligne de communication entre les détecteurs et la console de gestion des alertes, mais il peut être également utilisé comme :

- Un système d'archivage des alertes générées par une variété de détecteurs, pour une analyse ultérieure,
- Un système de corrélation des événements qui permet d'effectuer certains mécanismes de corrélation sur les alertes collectées par les différents détecteurs, pour ensuite générer des alertes plus complètes et globales,
- Une interface graphique qui permet la visualisation en temps réel des événements reportés par les différents détecteurs,
- Un format de données commun qui facilite la communication entre : utilisateur, vendeur, groupe de réponse, etc.

2.3. Modèle coopérative (Détection d'intrusions coopérative)

La détection d'intrusions coopérative permet à plusieurs IDS et autres outils de sécurité de coopérer ensemble.

L'objectif de cette coopération peut être l'utilisation de méthodes de détection différentes, par exemple la détection par signatures et la détection comportementale qui peuvent être complémentaires. On peut aussi utiliser plusieurs IDS basés sur la même approche, mais de constructeurs différents.

Un autre objectif de la détection d'intrusions coopérative, qui concerne le cas de la détection d'intrusions réseau, est d'obtenir une description de haut niveau des attaques en distribuant plusieurs IDS dans ce réseau tel que chacun sera associé à une partie du réseau. En plus des IDS, d'autres analyseurs peuvent être pris en compte dans la détection d'intrusions coopérative tels que les analyseurs réseaux et les scanneurs de vulnérabilités afin de permettre de corréler les alertes et les informations contextuelles telles que la topologie et la cartographie.

Cependant, XML est limité à une représentation syntaxique. Étant donné que cette représentation est dénuée de sémantique, chaque système doit interpréter et implémenter le modèle de données via un programme ce qui peut facilement générer des interprétations différentes par rapport aux mêmes données. Un formalisme logique s'impose alors. Dans ce cas, la logique propositionnelle n'est pas vraiment adéquate, car elle ne permet pas de représenter les informations de manière structurée. D'où la nécessité d'aller au delà de cette logique en termes d'expressivité.

Nous proposons alors de considérer un fragment de la logique du premier ordre, à savoir les logiques de description ou DLs (pour Description Logics). Le choix de telles logiques est justifié tout d'abord par le fait qu'elles conviennent à la représentation des informations structurées. De plus, on dispose actuellement d'un nombre considérable de logiques de description variant en termes d'expressivité, et pour lesquelles la complexité du raisonnement est bien connue. En outre, de nombreux raisonneurs en DLs ont été développés tel que Fact + + [51]. La plupart de ces raisonneurs utilisent des techniques d'optimisation sophistiquées. Les logiques de description sont bien

adaptées pour représenter des informations structurées et de plus, elles garantissent généralement la décidabilité du raisonnement.

3. Conclusion

Nous avons vu dans le deuxième chapitre que les problèmes majeurs des systèmes de détection d'intrusions sont : le taux très élevé de fausses alertes et les faibles taux de détection.

Pour cela, l'utilisation de plusieurs systèmes de détection d'intrusions et d'autres outils de sécurité peut améliorer le taux de la détection et diminuer les fausses alertes, mais cela nécessite l'existence d'un mécanisme de communication entre ces différents outils.

Dans cette optique, plusieurs efforts de standardisation ont été faits pour assurer l'interopérabilité entre plusieurs systèmes de détection d'intrusions.

Le modèle CIDF est le premier effort de standardisation, malgré son échec il a laissé un bon vocabulaire et un modèle de référence dans le domaine de la détection.

L'échec du CIDF a motivé le lancement du projet IDMEF qui a abouti à un standard de représentation des alertes, largement intégré dans les outils de nos jours.

Enfin, nous avons vu que pour améliorer les résultats de la corrélation d'alertes, il ne faut pas se limiter aux informations fournies dans les alertes. D'autres informations contextuelles sont nécessaires, telles que, la topologie et la cartographie des systèmes surveillés, et des informations sur les détecteurs, les vulnérabilités, et les attaques et c'est ce qu'on appel la détection d'intrusion coopératif.

Chapitre IV :

LA COOPERATION DES SYSTEMES DE DETECTION D4INTRUSIONS

1. Introduction

L'utilisation intensive des systèmes de détection d'intrusion a mis en évidence le problème de la gestion d'un grand nombre de flux d'alertes générés par ces systèmes. Toutes ces alertes ont une grande chance d'être des fausses alarmes. De plus, la majorité des systèmes de détection d'intrusion utilisent leur propre format d'alerte, ce qui a rendu difficile la construction d'une image globale à partir de l'analyse des alertes provenant de plusieurs détecteurs hétérogènes. L'inconvénient majeur des systèmes de détection d'intrusion est : « de ne pas être capable d'analyser et réagir devant un énorme volume d'alertes, générées dans des formats différents, et avec un taux de fausses alertes relativement élevé ».

A cet effet pour palier a cette problématique, une proposition d'un formalisme logique de représentations des informations contextuelles en détection d'intrusions est présentée. Par ailleurs, une explication de l'approche proposée pour la coopération des systèmes de détection d'intrusions ainsi qu'un algorithme de diminution de fausses alertes est suggéré

2. Représentation des informations contextuelles en détection d'intrusion

Les analyseurs utilisés en détection d'intrusions coopérative ne sont pas totalement fiables, souvent des conflits surviennent [52]. Par exemple, il est possible qu'un IDS détecte une tentative d'attaque contre un serveur IIS, alors que l'analyseur réseau indique que le serveur web est un Apache. Dans une telle situation, il est indispensable de gérer ces incohérences afin d'exploiter au mieux la coopération entre ces outils.

Les informations échangées en détection d'intrusions coopérative impliquent tout d'abord les alertes générées par les IDS. Pour la description d'alertes, nous nous sommes basés sur le format IDMEF, qui représente le standard actuel des formats d'alertes.

Pour traduire l'IDMEF en logique de description, nous utilisons une TBox. Cette TBox comprend des axiomes de définition ainsi que des axiomes d'inclusion.

2.1. Modélisation de la base de connaissance :

La modélisation de la base de connaissance passe par trois étapes : la première étant de conceptualiser une TBOX, la deuxième étant de proposer une ABOX, et la troisième consiste à utiliser l'inférence. Dans notre cas, nous allons modéliser deux bases de connaissances différentes , la première étant pour les IDS et dans ce cas on associera plusieurs Abox pour une même Tbox, et la deuxième étant la base de connaissances coopérative qui inclura la Tbox modélisée au niveau de chaque IDS en ajoutant d'autres axiomes, on nommera cette TBOX comme TBOX coopérative pour plus de facilité de description .

2.1.1. La TBOX

2.1.1.1. Les concepts rentrants dans la description des deux TBOX :

Id_alert : c'est l'identifiant de l'alerte

Id_alert ⊆ T

Time : c'est le concept heur

Time ⊆ T

Analyser : le SDI ayant généré cette alerte

Analyser ⊆ T

Additional_Data : ce champ peut comprendre tout type d'informations en dehors des informations précédentes, il garantit l'extensibilité du modèle

Additional_Data ⊆ T

Assessment: indique par exemple la gravité de l'attaque

Assessment ⊆ T

NameAttack: c'est le nom de l'attaque contenue dans l'alerte

NameAttack ⊆ T

Target: c'est la description de la victime

Target ⊆ T

2.1.1.2. Les propriétés rentrantes dans la description des deux TBOX :

date_creation : relie le concept Alert au concept Time.

date_creation ⊆ ∀ date_creation.Time

hasAdditionaldata: relie le concept Alert au concept Additional_Data.

hasAdditionaldata ⊆ ∀ hasAdditionaldata.Additional_Data

hasAnalyser: relie le concept Alert au concept Analyser.

hasAnalyser ⊆ ∀ hasAnalyser.Analyser

hasAssessment: relie le concept Alert au concept Assessment.

hasAssessment ⊆ ∀ est_evaluer.Assessment

hasId : relie le concept Alert au concept Id_alert.

hasId ⊆ ∀ hasId .Id_alert

hasTarget: relie le concept Alert au concept Target.

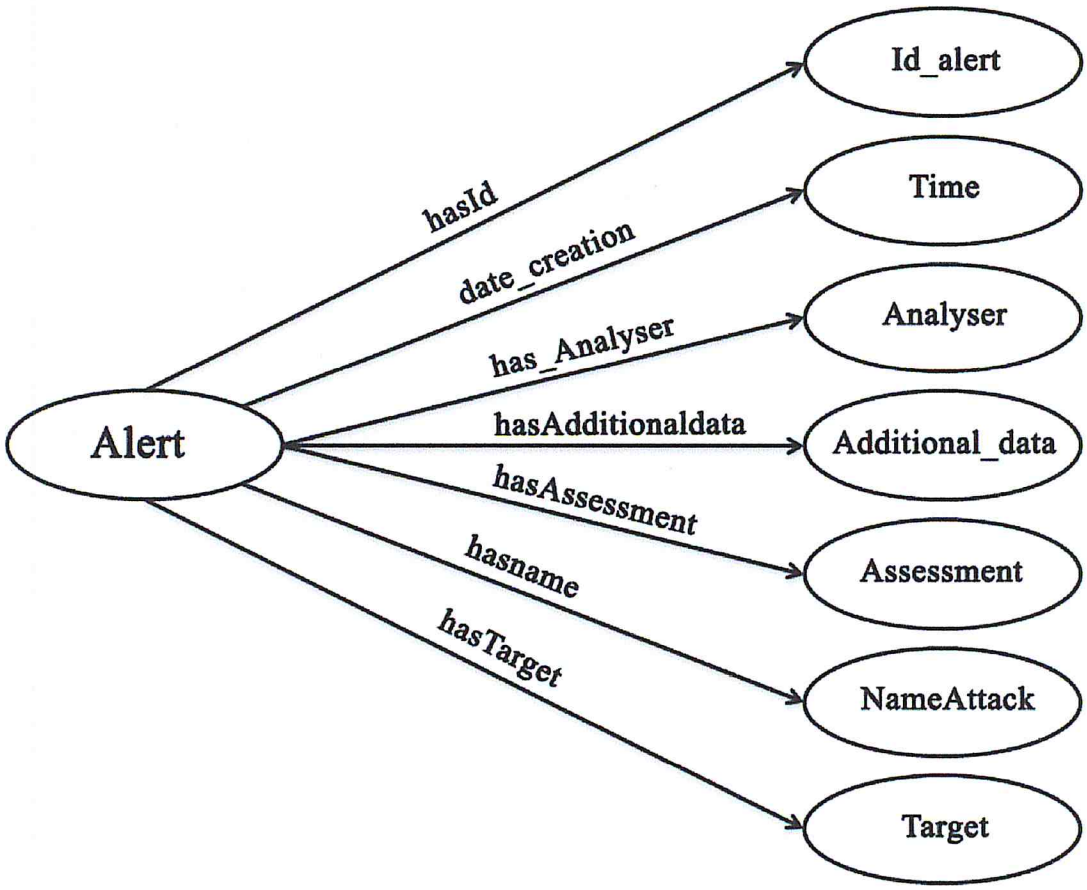
hasTarget ⊆ ∀ hasTarget.Target

hasName : relie le concept Alert au concept NameAttack.

hasName ⊆ ∀ hasName. NameAttack

Après avoir définit les concepts et les propriétés on pourra définir le concept Alert :

Alert : une alerte admet un seul identifiant, un seul champ « date_creation » de type « time », et un seul élément « Analyser ».
De plus, une alerte contient un seul nom d'attaque et une seule cible
En outre, une alerte admet un élément « Assesment » et un ou plusieurs données supplémentaires « Additional_Data ».



Alert ⊆ hasId ⋂ =1 hasId

⋂ date_creation ⋂ =1 date_creation

⋂ hasAnalyser ⋂ = 1 hasAnalyser

⋂ hasName ⋂ =1 hasName

⋂ hasTarget ⋂ =1 hasTarget

⋂ hasAssessment ⋂ =1 hasAssessment

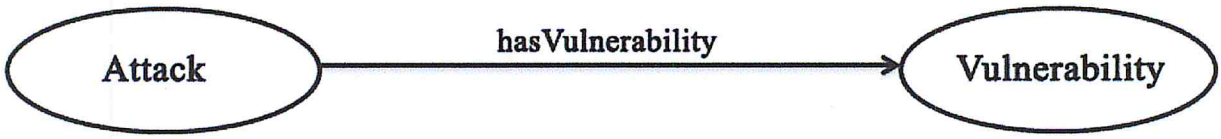
⋂ hasAditionaldata

2.1.1.3. Les concepts et propriétés rentrants dans la description de la TBOX coopérative:

En plus des concepts et des propriétés définis ci-dessus, la deuxième base de connaissance qui est appelé coopérative inclus les concepts et les propriétés suivantes :

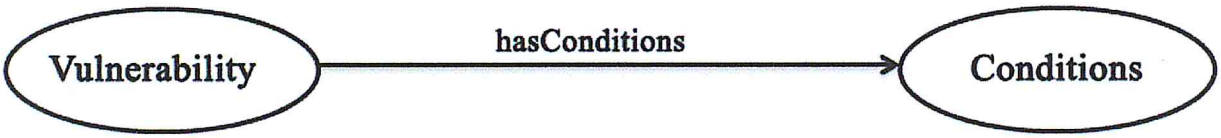
2.1.1.3.1. Les concepts

Attaque: c'est le concept attaque qui contient des attaque renferment des informations sur les vulnérabilités qu'ils touchent.



Attaque ⊆ ∃ hasVulnerability.Vulnerability

Vulnerability: c'est les vulnérabilités visées par les attaques.



Vulnerability ⊆ ∃ hasCondition.Conditions

Condition: c'est les conditions des systèmes pour qu'il soit vulnérable à une attaque donné.

Condition ⊆ T

2.1.1.3.2. Les propriétés

hasVulnerability : relie le concept Attaque au concept Vulnerability.

hasVulnerability ⊆ ∀ hasVulnerability.Vulnerability

hasCondition : relie le concept Alert au concept Nomattaque.

hasCondition ⊆ ∀ hasCondition.Condition

2.1.2. La ABOX

Comme on l'avait précisé auparavant, notre système contiendra plusieurs ABOX d'une même TBOX, ou chaque Abox appartient a la base de connaissance de L'IDS cette dernière contiendra les différentes alertes détectees par cet IDS contrairement à la ABOX de la base de connaissance coopérative qui contient toutes les alertes détecté par tous les IDS du système. En outre, la ABOX contiendra des informations récoltées de différentes sources de détections et des informations sur les attaques connues et les vulnérabilités que ces attaques visent.

2.1.3. L'inférence

Dans notre cas, l'inférence se fait au niveau factuel c'est-à-dire que l'inférence se fera entre les individus. Nous expliquerons plus en détail le mécanisme d'inférence dans le chapitre qui suit.

3. La proposition du système

L'approche consiste à faire coopérer plusieurs systèmes de détection d'intrusion sachant que les IDS ont des méthodes de détection différentes et que chaque IDS a sa propre méthode de détection. Le formalisme d'alerte généré étant le même pour tous les IDS.

Nous avons proposé une coopération entre IDS de telle manière que chaque IDS possède sa propre base de connaissance qui se compose de la TBOX définit dans les sections (2.1.1.1 et 2.1.1.2). Ces IDS sont reliés à une base de connaissance coopérative dont la TBOX contient les mêmes concepts et les mêmes propriétés que celle des bases de connaissance des IDS en plus des concepts et des propriétés définit dans la section (2.1.1.3).

La figure 4.1 montre l'architecture du système :

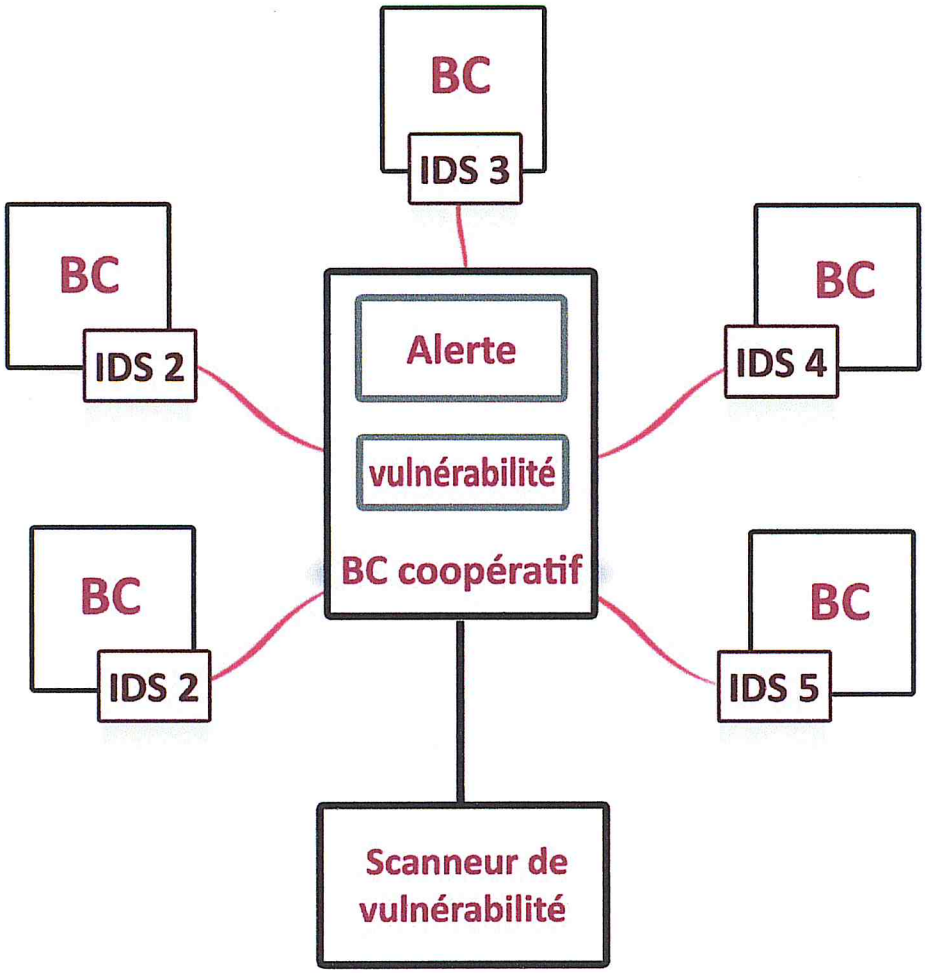


Figure 4.1 l'architecture du système

Le processus de travail définit la liaison de plusieurs IDS à une base de connaissance coopérative cette base contient toutes les alertes générées par les différents IDS, et les informations collectées d'autres analyseurs et des scanneurs de vulnérabilités.

Quand un des IDS détecte une attaque : il génère une alerte exprimé en logique de description, puis il suit les étapes suivantes :

1ere étape : l'IDS compare l'alerte générée avec sa base de connaissance par rapport au nom d'attaque et la cible. Si cet IDS trouves dans l'alerte générée la même cible et le même nom d'attaque que dans une alerte dans sa base de connaissance, celle la sera déclarées comme une attaque.

Après comparaison si l'IDS ne trouve pas dans sa base de connaissance, la même

cible et le même nom d'attaque alors il passe à la deuxième étape.

2eme étape : l'alerte est envoyée vers la base de connaissance coopérative pour chercher s'il y'a une même cible et un même nom d'attaque que celle dans l'alerte générée.

Si le même nom d'attaque et la même cible ont été reconnues alors l'alerte est ajouté au niveau de la base de connaissance de l'IDS générant l'alerte.

Si l'alerte ne se trouve pas ni dans la base de connaissance de l'IDS ni dans la base de connaissance coopérative la question se pose es-ce que c'est une fausse alerte ou es-ce que l'alerte est considéré comme une nouvelle attaque ?

La méthode proposée sera illustrée dans ce qui suit

4. La diminution des fausses alertes

Dans cette section nous allons proposer une solution pour diminuer les fausses alertes. Dans ce cas, ou la même alerte (nom d'attaque et cible) n'est trouvée ni dans la base de connaissance de l'IDS qui a généré l'alerte ni dans la base de connaissance coopérative alors que doit-on faire ?

Dans le cas ou l'un des deux champs (nom d'attaque, cible) est égale à un champ d'une alerte dans la base de connaissance, l'autre étant différent on procède comme suit :

On compare tout d'abord le nom de l'attaque contenu dans l'alerte avec le champ des attaques connue dans la base de connaissance coopérative, si on trouve l'attaque on extrait les conditions du système vulnérable a cette attaque et on le compare au champ « hasAdditionalData » qui contenant les informations sur le système attaqué. Si ces conditions sont identiques au champ « hasAdditionalData » dans l'alerte alors celle-ci sera déclarée comme une attaque. Si ce n'est pas le cas alors on vérifier le champ « évaluation » si c'est « high » alors l'alerte est une attaque si c'est « low» c'est une fausse alerte.

Dans le cas ou on ne trouve pas le nom d'attaque qui figure dans les alerte connue alors le champ « hasAdditionalData » sera comparé avec toutes les conditions des

systèmes liés à des vulnérabilités qui figure dans la base de connaissance coopérative.
Si un ensemble de conditions lié à une vulnérabilité est identique au champ « hasAdditionalData » alors l'alerte sera déclarée comme attaque.
Sinon on procède à la vérification du champ « évaluation » si c'est « high» alors c'est une attaque si c'est « low» c'est une fausse alerte.
Dans un autre cas ou ni le nom de l'attaque ni la cible ne figure pas dans les alertes stockées au niveau de la base de connaissance coopérative alors cette alerte sera considérée comme une nouvelle attaque.
Dans tous les cas ou l'alerte est déclarée comme une attaque, celle-ci sera ajoutée dans la base de connaissance coopérative ainsi que dans la base de connaissance de l'IDS qui a généré cette alerte.
Cette partie de traitement se fait au niveau de la base de connaissance coopérative.

L'algorithme de la diminution des fausses alertes

Dans le cas ou une alerte n'est pas trouver dans notre système nous allons procéder selon l'algorithme que nous proposant :

Si ((nom d'attaque de l'alerte généré est le même que le nom d'attaque dans une autre alerte dans la base de connaissance coopérative **ET** que la cible est différente) **OU** inversement) alors

On prend le nom d'attaque contenu dans l'alerte et le comparer avec les attaques connues répertoriés dans la base de connaissance coopérative.

- Si le nom d'attaque est trouvé alors on compare le champ « Addittional_Data » avec les conditions du système qui sont vulnérables à ce nom d'attaque.
    - Si condition = Additional_Data alors déclarer l'attaque et ajouter l'alerte au deux bases de connaissance (coopérative, et celle de l'IDS).
    - Sinon vérifier le champ « assessmant »
        - si assessment = high alors déclarer l'attaque et ajouter l'alerte au deux bases de connaissance (coopérative, et celle de l'IDS).
        - Sinon c'est une fausse alerte
- Sinon on compare le champ « Addittional_Data » avec chaque groupe de conditions lié à une vulnérabilité
    - Si condition = Additional_Data alors déclarer l'attaque et ajouter l'alerte au deux base de connaissance (coopérative, et celle de l'IDS).
    - Sinon vérifier le champ « assessmant »
        - si assessment = high alors déclarer l'attaque et ajouter l'alerte au deux base de connaissance (coopérative, et celle de l'IDS).
        - Sinon c'est une fausse alerte

Si aucun des deux paramètres (nom d'attaque et cible) ne sont trouvés dans la base de connaissance coopérative alors on considèrera que c'est une nouvelle attaque et ajouter l'alerte au deux bases de connaissance (coopérative, et celle de l'IDS).

5. Conclusion

Un vocabulaire basé sur les logiques de description dans le cadre de la détection d'intrusions coopérative est proposé. Ce vocabulaire servira exclusivement pour permettre l'échange des informations entre plusieurs systèmes de détection d'intrusion, à travers une base de connaissance coopérative. Une présentation d'un système de détection d'intrusions coopératif et son architecture sont mise en place, ce qui permettra une coopération entre les différents IDS. Pour une meilleure gestion des fausses alertes un algorithme de diminution de fausses alertes est établie.

Chapitre V :

LA REALISATION DU SYSTEME

1. INTRODUCTION

Le but de ce chapitre est d'apporter une description claire du système implémenté. Ce chapitre se présente sous forme de trois sections.

Dans la première section nous fournissons un aperçu sur les langages et les outils utilisés pour le développement du noyau de l'application, puis dans la seconde section nous introduisons le module d'inférence et les mécanismes de raisonnement et d'enrichissement d'une base de connaissances en logique de description. Dans la troisième section nous présentons le module d'interrogation ainsi que les outils pour le mettre en œuvre.

Nous parlons en fin de l'implémentation de notre système et une proposition pour la diminution des fausses alertes.

2. BASE DE CONNAISSANCE

Ce module dédié à la base de connaissance représente le cœur de notre application. La base de connaissances est représentée en logique de description. Cette dernière contiens l'ensemble des concepts et des propriétés (Tbox), des individus (Abox) et les règles d'inférences qui régissent le tout.

La question à poser est la suivante : Quels sont les outils qui nous permettent de construire une base de connaissances en logique de description ?

2.1. Le sous langage OWL- DL

OWL DL est une déclinaison d'OWL, permettant une expressivité bien plus importante. OWL DL est fondé sur la logique descriptive (d'où son nom, OWL Description Logics), un domaine de recherche étudiant la logique, et conférant donc à OWL DL son adaptation au raisonnement automatisé. Malgré sa complexité relative face à OWL Lite,

OWL-DL garantit la complétude des raisonnements (toutes les inférences sont calculables) et leur décidabilité (leur calcul se fait en une durée finie).

OWL-DL est une version décidable du langage informatique OWL. Ce langage correspond à la logique de description *SHOIN(D)* [2].

2.2. L'éditeur protégé

L'éditeur Protégé, développé par l'Université de Stanford en collaboration avec l'Université de Manchester, est le standard de facto pour la création et l'édition d'ontologies OWL.

Cet éditeur offre, avec une interface graphique très agréable et facile à prendre en main tous les outils pour créer et manipuler facilement des ontologies dans divers formats de représentation. Pour être plus claire ce n'est pas un outil dédié à OWL, mais un éditeur hautement extensible : par exemple le support d'OWL est possible grâce à un plugin dédié. Parmi les autres éditeurs d'ontologies, moins répandus, nous pouvons citer KAON2 , Swoop et Ontolingua[53] .

Notons que la version utilisée de protégé est 3.5 [54].

2.2.1. la base de connaissance au format OWL

L'édition de la base de connaissance se fait en construisant sa Tbox, c'est-à-dire la construction de l'ensemble des concepts et des propriétés défini dans le formalisme de la logique de description en utilisant protégé. Le résultat de ce travail est un fichier OWL appelé : base_connaissance.owl

Exemple

La relation Alert selon le modèle IDMFE est exprimée en logique de description comme ceci :

Alert ⊆ hasId ⋂ =1 hasId

⋂ date_creation ⋂ =1 date_creation

⋂ hasAnalyser ⋂ = 1 hasAnalyser

⋂ hasName ⋂ =1 hasName

⋂ hasTarget ⋂ =1 hasTarget

⋂ hasAssessment ⋂ =1 hasAssessment

⋂ hasAditionaldata

Et ou hasId, date_creation, hasAnalyser, hasName, hasTarget, hasAssessment, hasAditionaldata sont des relations binaires tel que :

- hasId (Alert, Id_alert)
- date_creation (Alert, Time)
- hasAnalyser (Alert, Analyser)
- hasName (Alert, NameAttack)
- hasTarget (Alert, Target)
- hasAssessment (Alert, Assessment)
- hasAditionaldata (Alert, Additional_Data)

Et où : Id_alert, Time, Analyser, NameAttack, Target, Assessment et Additional_Data sont des concepts.

Pour générer un fichier OWL qui contiendra notre base de connaissances.

Nous suivrons les étapes suivantes :

Étape 1 : En utilisant protégé nous définissons d'abord l'ensemble des concepts, c'est-

à dire définir les classes : Id_alert, Time, Analyser, NameAttack, Target, Assessment et Additional_Data et bien sur la relation Alert (devenue concept).

Dans la figure (5.1) :

- (1) et (2) sont les composants utilisés pour définir la TBOX dans protégé.
- (3) (Onglet individual) est le composant utilisé pour enrichir la Abox à partir de protégé.
- (4) représente les concepts (classes) : Id_alert, Time, Analyser, NameAttack, Target, Assessment et Additional_Data représenté dans protégé.
- (5) représente le concept Alert.

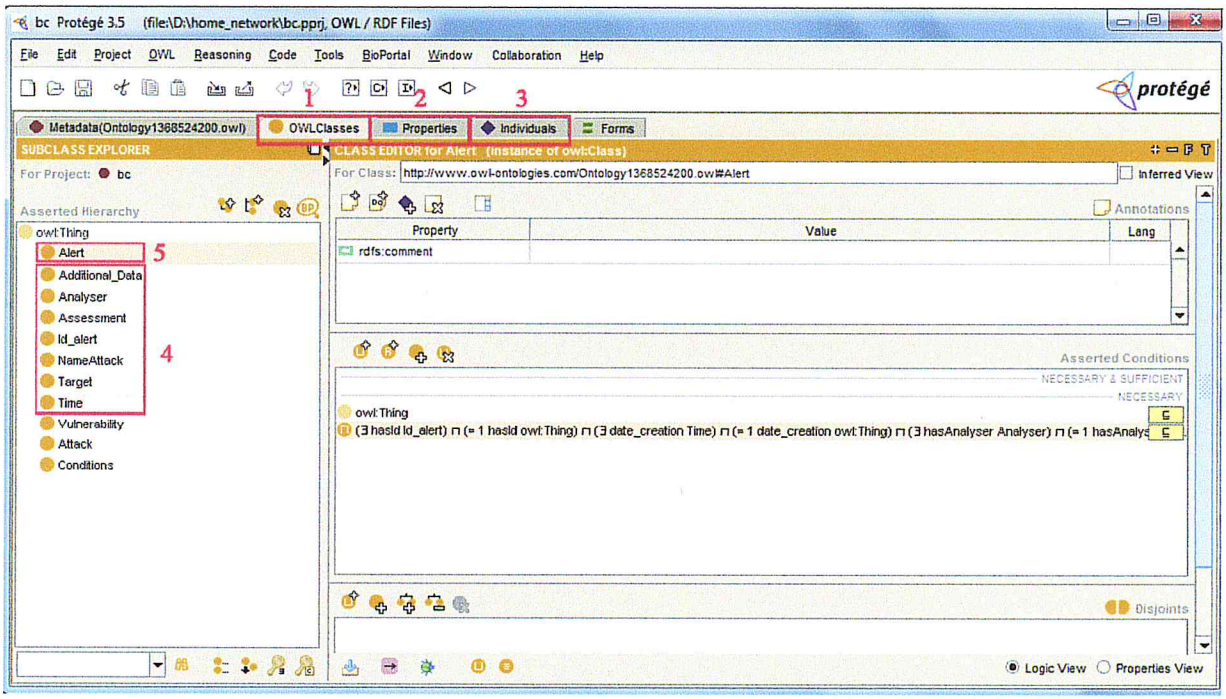


Figure 5.1: Onglet OWL classes dans protégé.

Étape 2 : Maintenant, nous définissons les relations, leurs rangs et leurs domaines, nous prenons la relation date_creation comme exemple, elle a pour rang : Alert et pour domaine : Time.

Dans la figure (5.2) :

- (1) représente la propriété : date_creation.
- (2) représente le domaine : Alert.
- (3) représente le rang : Time.

De la même façon, nous définissons les autres relations binaires pour les autres relations.

.

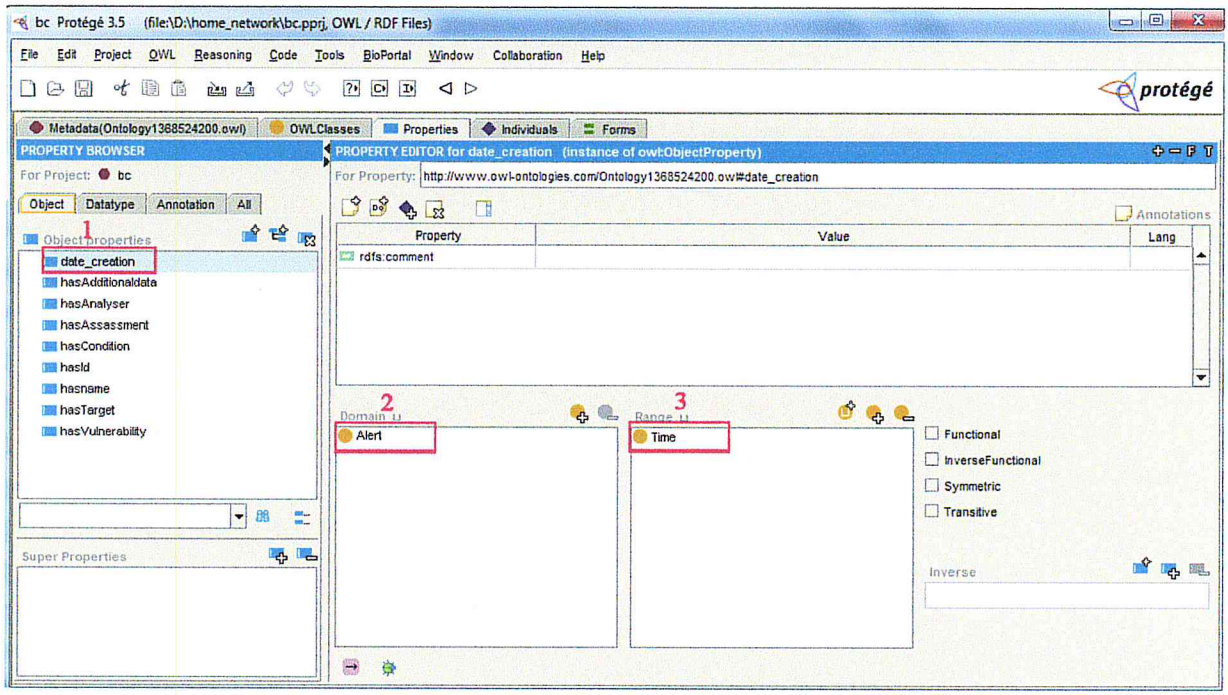


Figure 5.2: Onglet Propriétés dans Protégé.

Étape 3 : Définir la restriction sur la classe Alert c'est-à-dire :

Alert ⊆ hasId ⋂ =1 hasId

⋂ date_creation ⋂ =1 date_creation

⋂ hasAnalyser ⋂ = 1 hasAnalyser

⋂ hasName ⋂ =1 hasName

⋂ hasTarget ⋂ =1 hasTarget

⋂ hasAssessment ⋂ =1 hasAssessment

⋂ hasAditionaldata

Dans la figure (5.3) :

- (1) représente l'éditeur de restrictions.
- (2) représente la restriction sur la classe Alert.

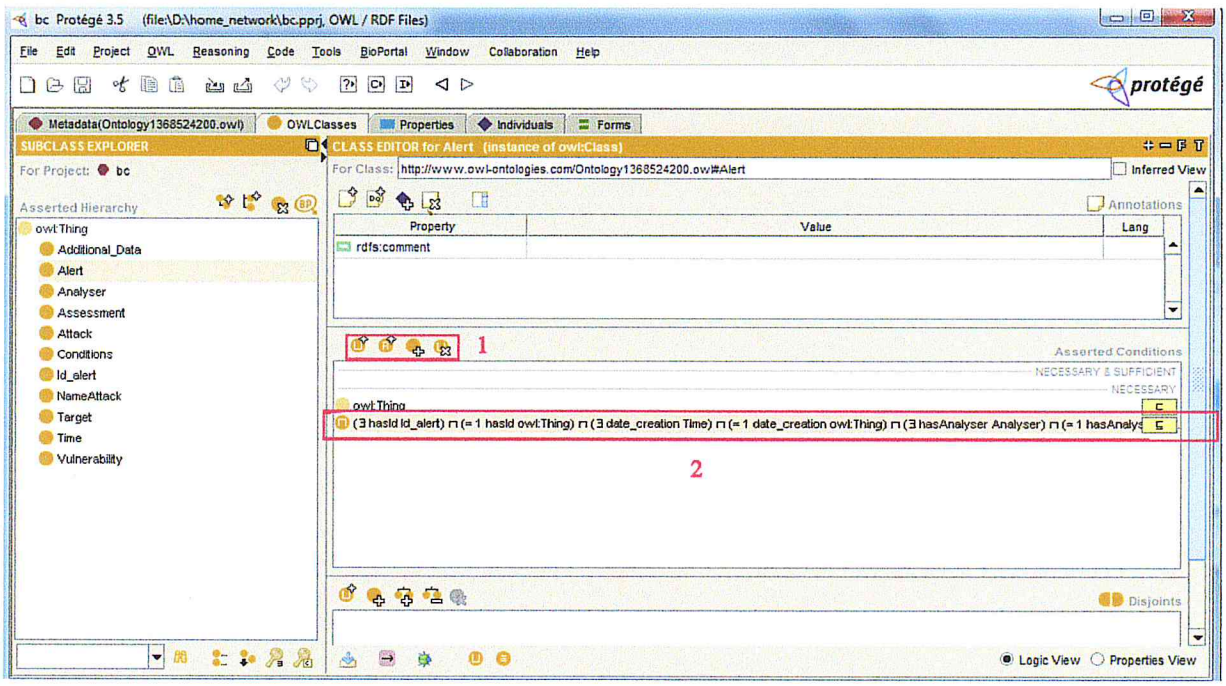


Figure 5.3: Onglet OWL Classes et les restrictions sur les classes dans protégé.

Rappelons que le fichier résultant est le noyau de notre application, un fichier qui porte l'extension OWL et qui contient une Tbox et une Abox.

3. L'inférence

Le fait que la base de connaissance possède une base théorique formelle (la logique de description) permet l'implantation de logiciels appelés moteurs d'inférence ou raisonneurs, qui sont capables de traiter une base de connaissance pour déduire des faits qui ne sont pas explicitement déclarés. C'est à dire, ils peuvent trouver des informations qui sont implicitement contenues dans une base de connaissances pour les rendre explicites. Ce processus s'appelle inférence ou raisonnement.

La principale tâche de raisonnement qu'un moteur d'inférence peut effectuer, est la fonction de subsomption.

La subsomption permet de savoir si une classe est une sous-classe d'une autre ou non. En effectuant cette tache sur toutes les classes de la TBOX de la base de connaissance, le moteur d'inférence peut construire une hiérarchie de classes inférée (en opposition à la hiérarchie de classes déclarées) dans laquelle toutes les relations super-classe/sous-classe sont explicitées.

Une autre tâche de raisonnement standard est la vérification de la cohérence de l'ontologie, qui permet de détecter s'il y a des classes qui ne sont pas cohérentes, c'est-à-dire, pour lesquelles il n'est pas possible de déclarer un individu sans avoir une contradiction logique. Cette tâche est très utile lors du processus de création d'une ontologie pour le débogage de celle-ci.

Les moteurs d'inférence peuvent utiliser ces tâches de raisonnement de base afin de fournir des services de raisonnement plus complexes, notamment :

- Recherche : Permet de trouver tous les individus qui sont des instances (directes ou indirectes) d'un concept donné ;
- Réalisation : Permet de trouver le concept le plus spécifique auquel appartient (directement ou indirectement) un individu donné.

Nous allons dans ce qui suit vous présenter le moteur d'inférence utilisé pour la réalisation de l'application

3.1. Moteur d'inférence jess

Jess est un moteur d'inférence très populaire crée par Sandia National Laboratories [55].

Probablement le meilleur moteur d'inférence sur le marché, sa puissance (il a été choisit pour être embarqué dans les systèmes informatiques des nouveaux destroyers américain DDG1000), ses mises à jour périodiques et son API java.

Ce moteur d'inférence possède un langage propre pour l'expression des connaissances sous forme de règles. Il peut être utilisé depuis Protégé (ou Protégé-OWL API) grâce à l'existence d'un pont (OWLJessBridge) qui permet de traduire un modèle d'ontologie dans le langage de Jess, d'exécuter les règles dans Jess et finalement de récupérer le résultat dans Protégé (ou dans Protégé-OWL API).

4. Module d'interrogation

Le module d'interrogation présente le mécanisme utilisé dans notre application pour réaliser des interrogations complexes sur la base de connaissances. Nous avons choisi d'interroger cette dernière en utilisant le langage SPARQL.

Depuis le 15 Janvier 2008, SAPRQL est devenu une recommandation dans le cadre de l'activité Web sémantique du W3C [56]. Par définition c'est un langage et un protocole de requêtes pour l'interrogation de méta données sous forme de fichiers RDF (Resource Description Framework) ou plus exactement un langage d'interrogation de triplets RDF, notons qu'un triplet RDF est une association : sujet, objet, prédicat.

Le langage SPARQL s'inspire clairement dans sa syntaxe et dans ses fonctionnalités du langage SQL. Il a aussi quelques traits de ressemblance mineurs avec Prolog.

SPARQL permet d'exprimer des requêtes interrogatives ou constructives :

- Une requête SELECT, de type interrogative, permet d'extraire du graphe RDF un sous-graphe correspondant à un ensemble de ressources vérifiant les conditions définies dans une clause WHERE.

- Une requête CONSTRUCT, de type constructive, engendre un nouveau graphe qui complète le graphe interrogé.

Il existe d'autres requêtes de type interrogatives :

- ASK va répondre par OUI ou NON à une question.
- DESCRIBE permet d'obtenir des informations détaillées sur la base de connaissance ou sur ces objets.

Dans notre application, seules les requêtes interrogatives de type SELECT ont été utilisées.

Pour le moteur de requêtes, nous utilisons ARQ un «query engine» utilisé dans le Jena framework, d'ailleurs nous avons travaillé avec l'API java de JENA dans le module d'interrogation [57].

Voici un exemple d'une requête SPARQL :

```
PREFIX ontology – name :_<< http://www.owl-
ontologies.com/Ontology1368524200.owl#>>
SELECT ?alert ?analyser from ontology-name
WHERE ?alert ontology-name : est_analyser
?Analyser.
```

L'exécution de cette requête nous donne les alertes et leur analyseur (en utilisant l'ontologie base_connaissance.owl).

5. Langage de programmation

Protégé propose aux développeurs une API java pour manipuler et travailler avec les ontologies OWL, il a été lui-même bâtis en utilisant JENA[57] (un frameWork proposant un cadre de travail pour la manipulation des ontologies à bas niveau en java), de ce fait le langage de programmation utilisé est le langage java.

5.1. Java

Java [16] est à la fois un langage de programmation informatique orienté objet et un environnement d'exécution informatique portable créé par James Gosling et Patrick Naughton employés de Sun Microsystems avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld. Le langage Java a la particularité principale que les logiciels écrits avec ce dernier sont très facilement portables sur plusieurs systèmes d'exploitation tels que Unix, Microsoft Windows, Mac OS ou Linux avec peu ou pas de modifications. C'est la plate-forme qui garantit la portabilité des applications développées en Java.

Le langage reprend en grande partie la syntaxe du langage C++, très utilisé par les informaticiens. Néanmoins, Java a été épuré des concepts les plus subtils du C++ et à la fois les plus déroutants, tels que l'héritage multiple remplacé par l'implémentation des interfaces.

Les concepteurs ont privilégié l'approche orientée objet de sorte qu'en Java, tout est objet à l'exception des types primitifs (nombres entiers, nombres à virgule flottante, etc.).

5.2. Environnement de développement java

Nous avons opté pour l'environnement de développement java Eclipse pour sa rapidité en temps d'exécution et sa simplicité en plus de sa disponibilité gratuite sur le net.

Eclipse est un projet, décliné et organisé en un ensemble de sous-projets de développements logiciels, de la Fondation__Eclipse visant à développer un environnement de production de logiciels libres qui soit extensible, universel et polyvalent, en s'appuyant principalement sur Java. Son objectif est de produire et fournir des outils pour la réalisation de logiciels, englobant les activités de programmation (notamment environnement_de_développement intégré et frameworks) mais aussi d'ATL recouvrant modélisation, conception, testing, gestion de configuration, reporting... Son EDI, partie intégrante du projet, vise notamment à supporter tout langage de programmation à l'instar deMicrosoft Visual Studio.

Bien que conçu initialement uniquement pour produire des environnements de développement, ses utilisateurs et contributeurs se sont rapidement mis à réutiliser ses librairies logicielles pour des applications clientes classiques. Cela a conduit à une extension du périmètre initial d'Eclipse à toute production de logiciel : c'est l'apparition du framework Eclipse RCP en 2004.

Figurant parmi les grandes réussites de l'OpenSource, Eclipse est devenu un standard du marché des logiciels de développement, intégré par de grands éditeurs logiciels et sociétés de services. Les logiciels commerciaux *Lotus Notes 8*, *IBM Lotus Symphony* ou *WebSphereStudio Application Developer* sont notamment basés sur Eclipse.

6. Choix de l'outil

Ici, nous faisons un récapitulatif sur les choix que nous avons fait pour la création de notre système.

Le tableau 5.1 résume ces choix. Le choix retenu est le suivant : Protégé 3.5 pour la création et l'édition d'ontologies,

Protégé-OWL API pour l'accès par programme à l'ontologie, Jess comme moteur d'inférence, SPARQL pour le langage de requête et Eclipse comme IDE JAVA.

Création/édition d'ontologies	Protégé 3.5
Accès par programme aux ontologies	Protégé-OWL API
Moteur d'inférence	Jess 7.0
Version d'OWL	OWL 1
IDE JAVA	Eclipse

Table 5.1: Choix d'outils et de langages pour la création de l'application.

7. Implémentation

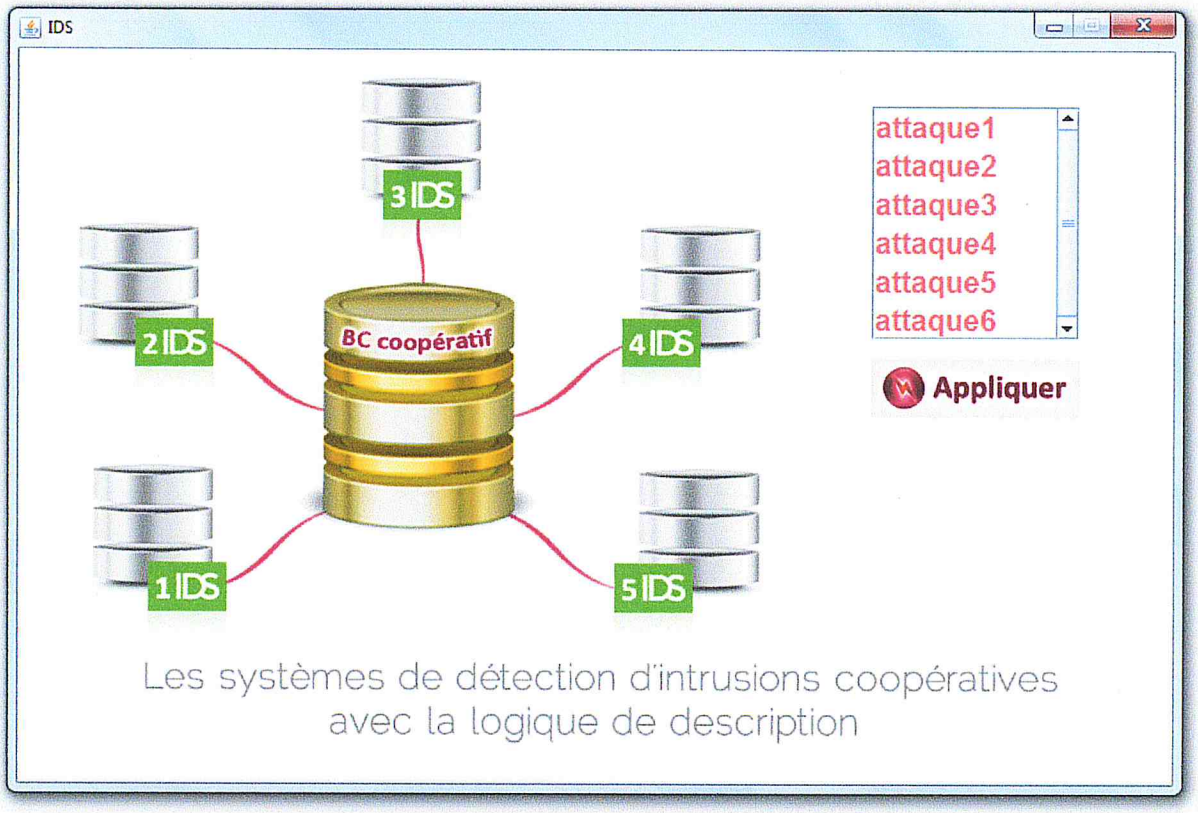


Figure 5.4: L'interface principale.

La figure 5.4 ci-dessus présente l'interface de l'application qui contient une liste d'attaques et qui vont être appliqués sur les 5 IDS relié a la base de connaissance coopérative afin de pouvoir faire la simulation et montrer la coopération entre les différents IDS.

Notre système contiendra 5 IDS qui utilisent des différentes méthodes de détection :

IDS1, IDS3, IDS5 : utilise l'approche comportementale.

IDS 2, IDS 4 : utilise l'approche basée connaissance.

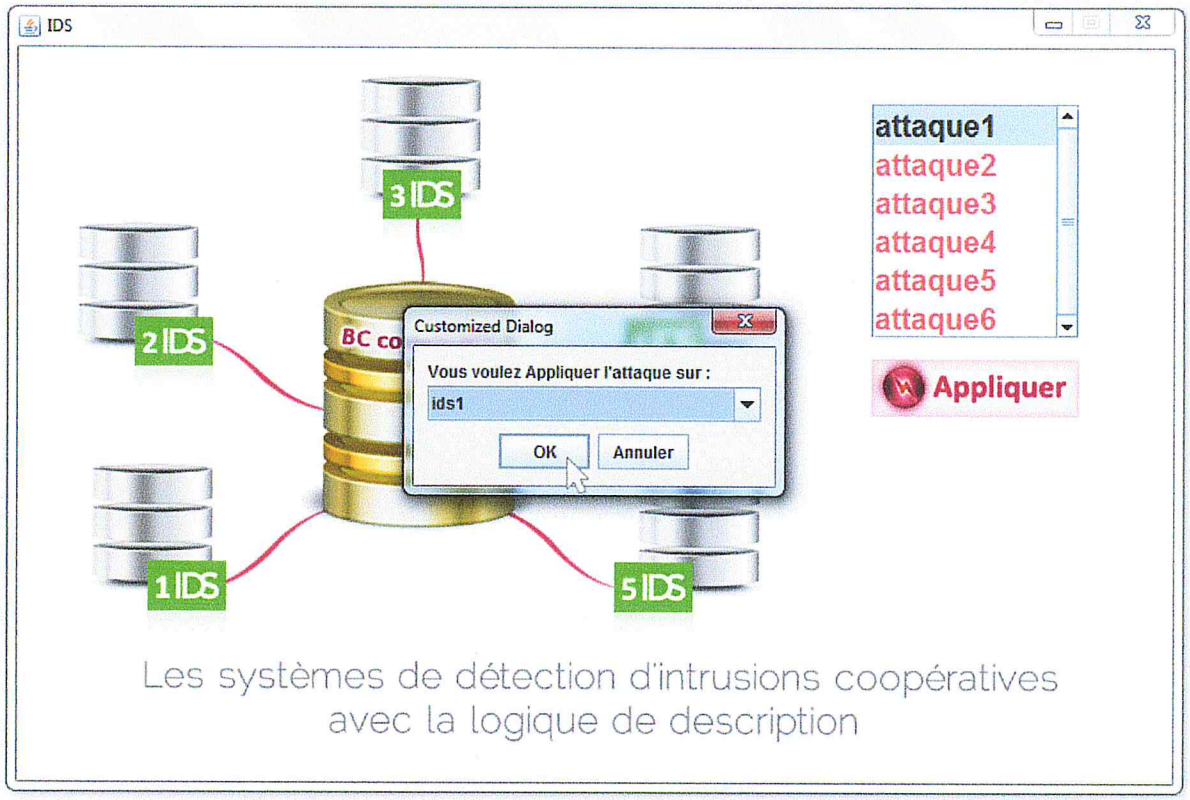


Figure 5.5: sélection de l'IDS.

Comme illustre sur la figure 5.5 on appliquera l'attaque « attaque 1 » sur l'ids1 qu'on a choisi.

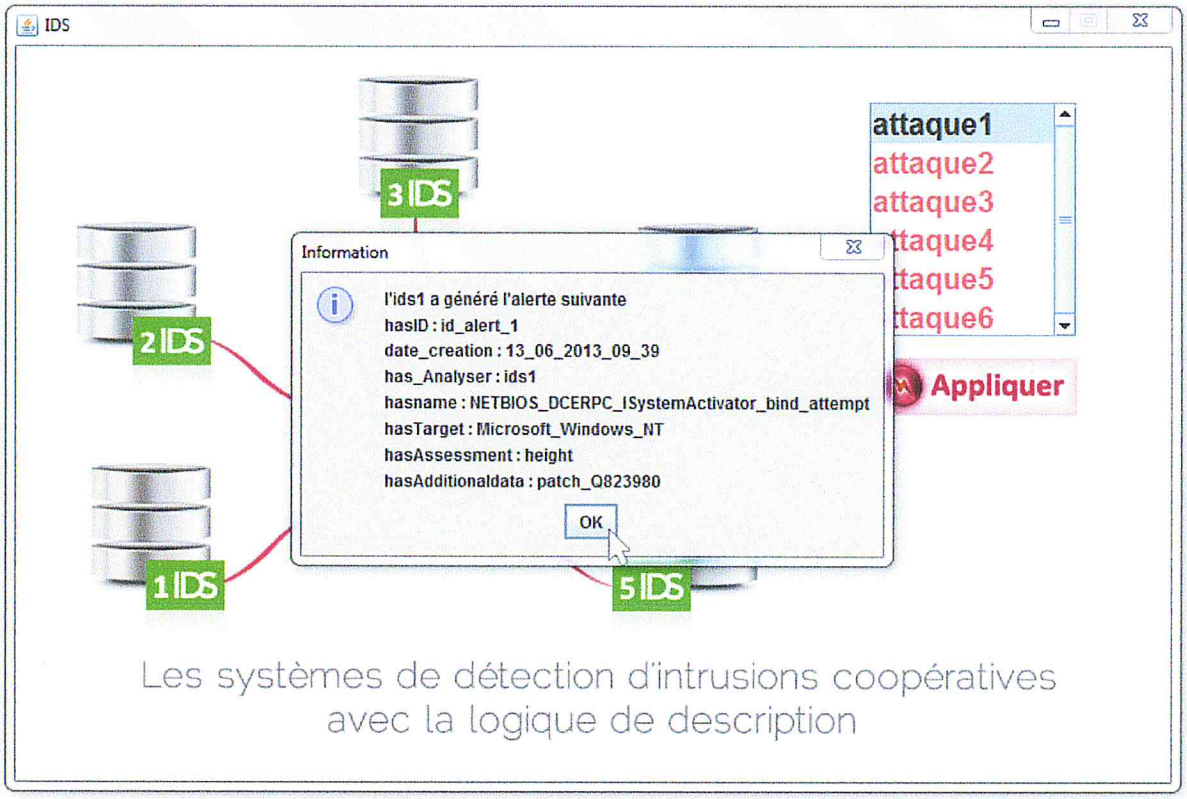


Figure 5.6: Format de l'alerte.

A partir des informations de l'attaque et avec la méthode de détection de l'IDS, l'alerte a été générée avec comme paramètre un identifiant, une date de création, l'analyseur qui a généré cette alerte, la cible visé, l'évaluation et des données supplémentaire.

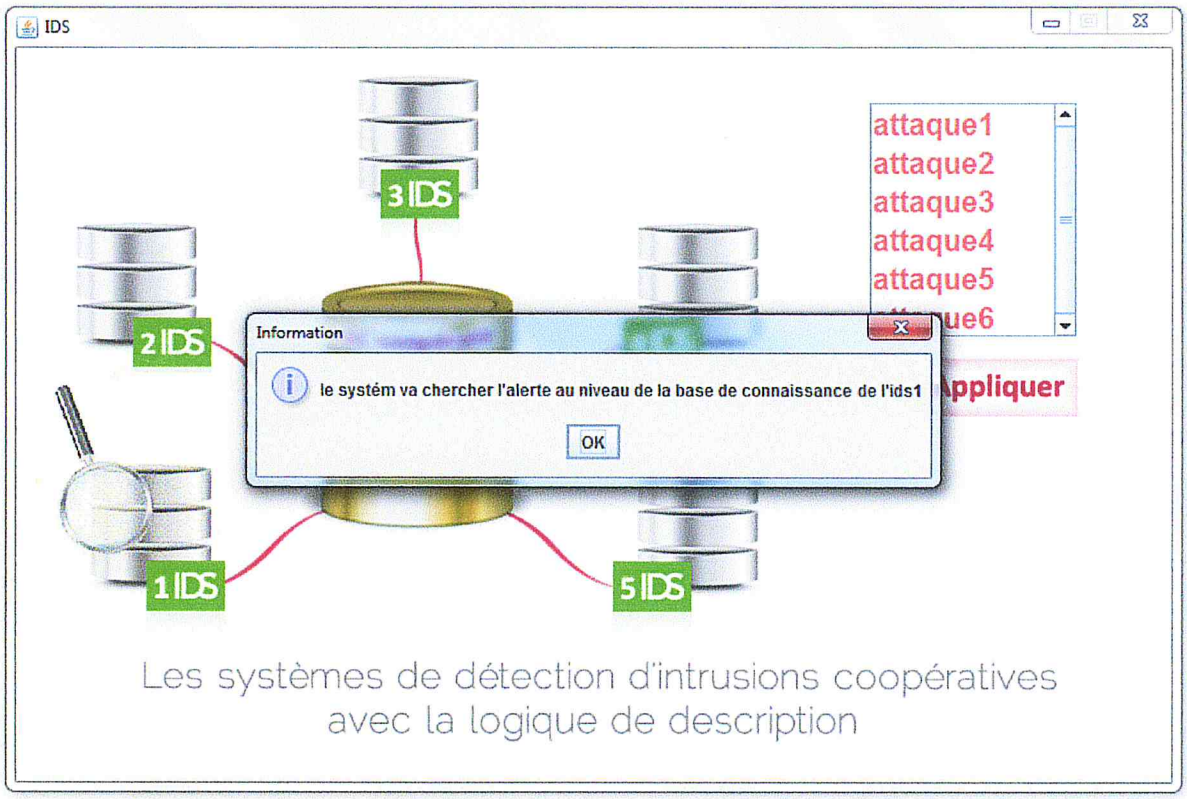


Figure 5.7: message de recherche.

Comme illustre la figure 5.7 la recherche va commencer au niveau de la base de connaissance de l'ids1.

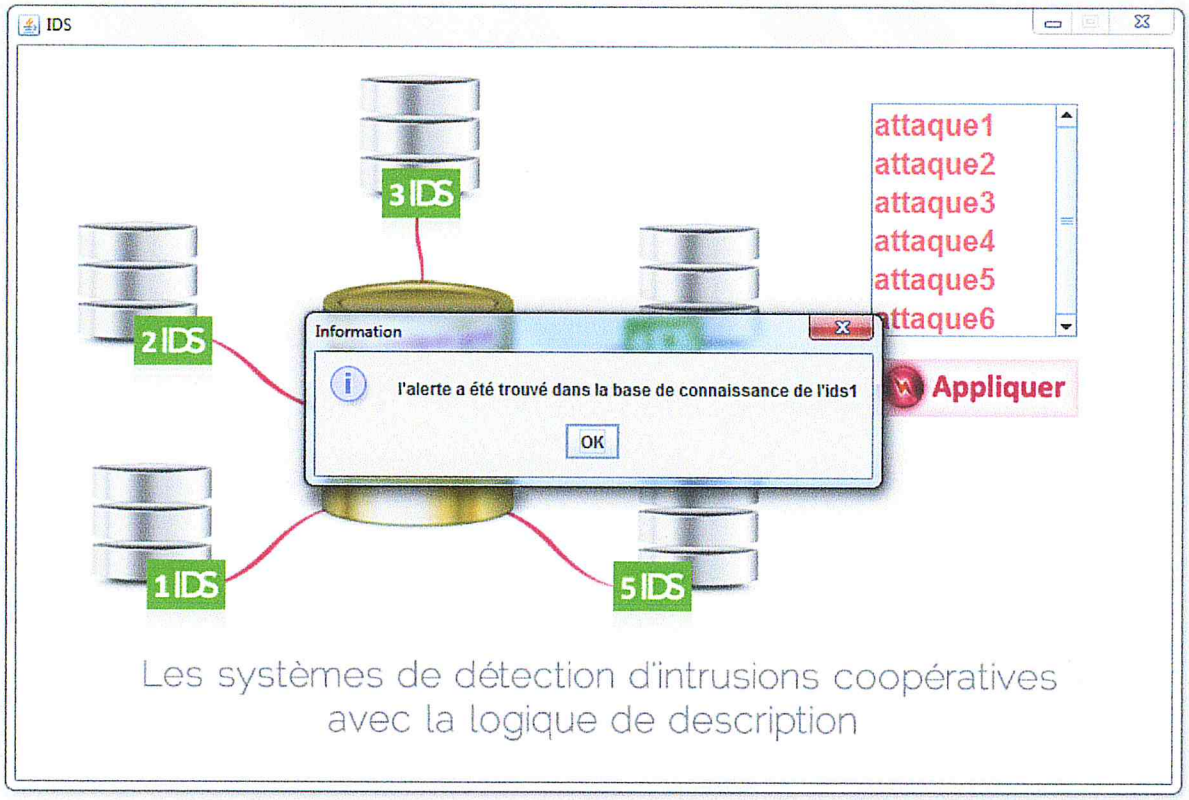


Figure 5.8: message d'information.

Un message indique que la recherche est finie et que l'alerte a été trouvée dans la base de connaissance de l'ids1.

Exemple 2:

Dans cet exemple nous illustrons le cas ou une alerte n'est pas trouver dans la base de connaissance de l'IDS et sera trouvé au niveau de la base de connaissance coopérative.

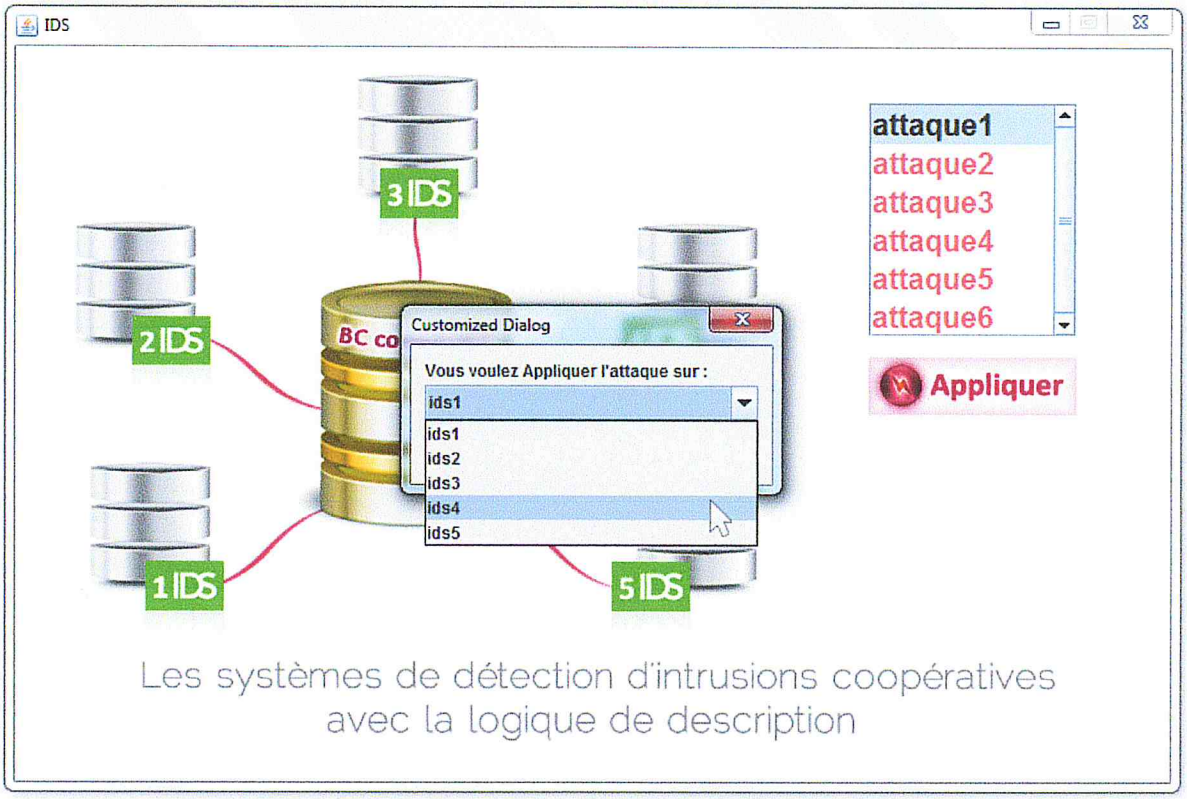


Figure 5.9: sélection de l'IDS.

Comme illustre la figure 5.9 on appliquera l'attaque « attaque 1 » sur l'ids4 qu'on a choisi.

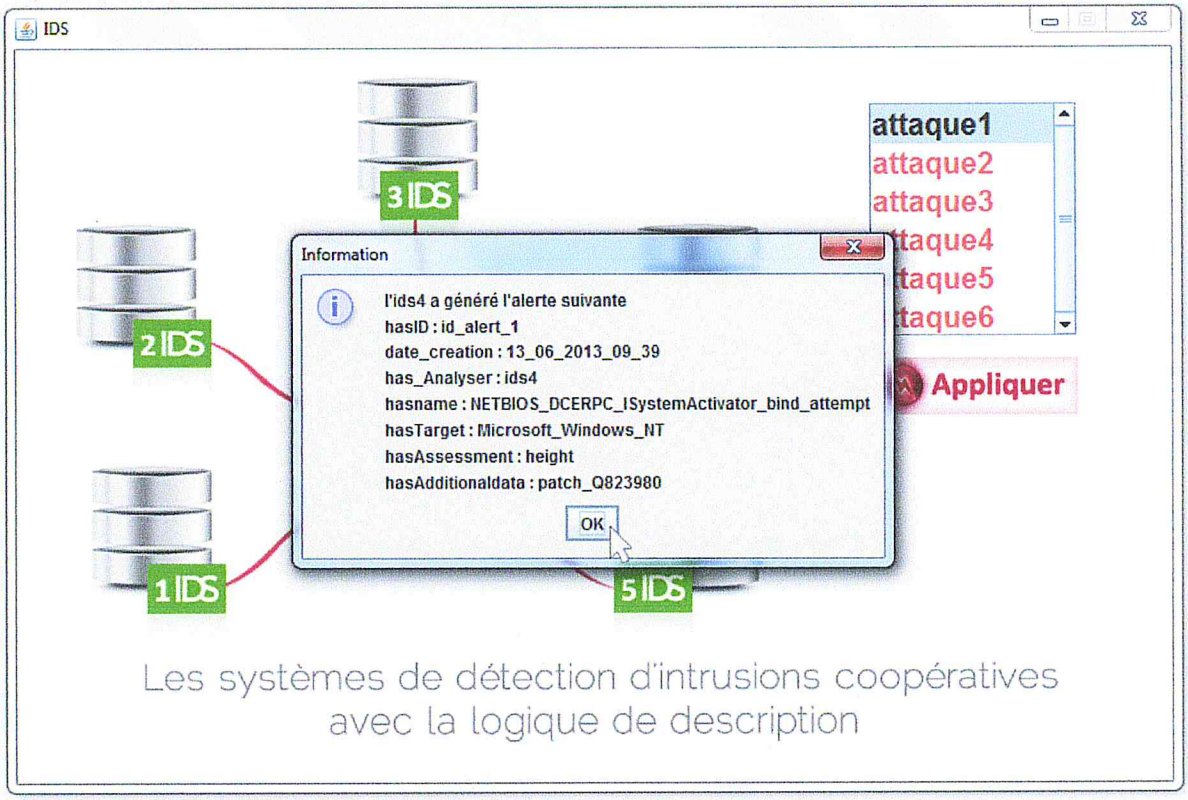


Figure 5.10: Format de l'alerte.

A partir des informations de l'attaque et avec la méthode de détection de l'ids4, l'alerte a été générée.

On remarque que malgré que la méthode de détection de l'ids1 prise dans l'exemple précédent est différente de la méthode de détection de l'ids4 prise dans cette exemple sauf que l'alerte générée possède la même structure que ça soit pour l'ids1 ou l'ids4.

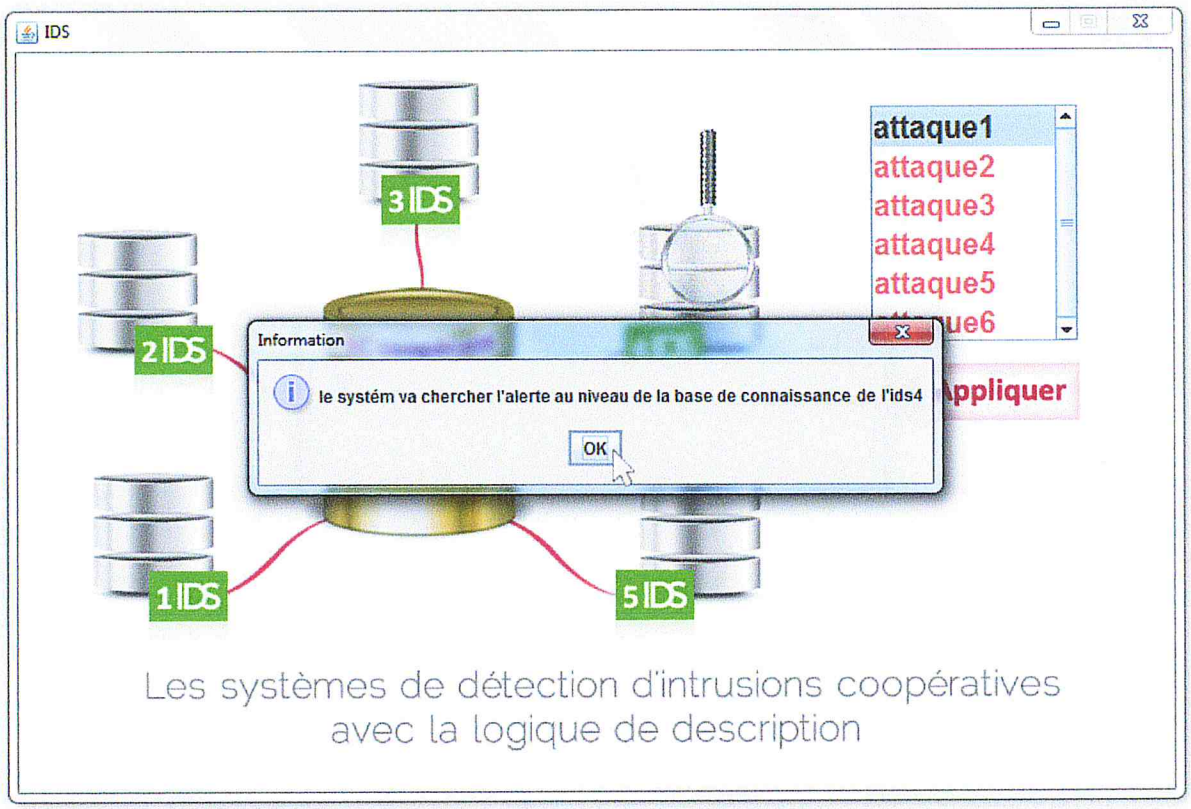


Figure 5.11: message de recherche.

Comme illustre la figure 5.11 la recherche va commencer au niveau de la base de connaissance de l'ids4

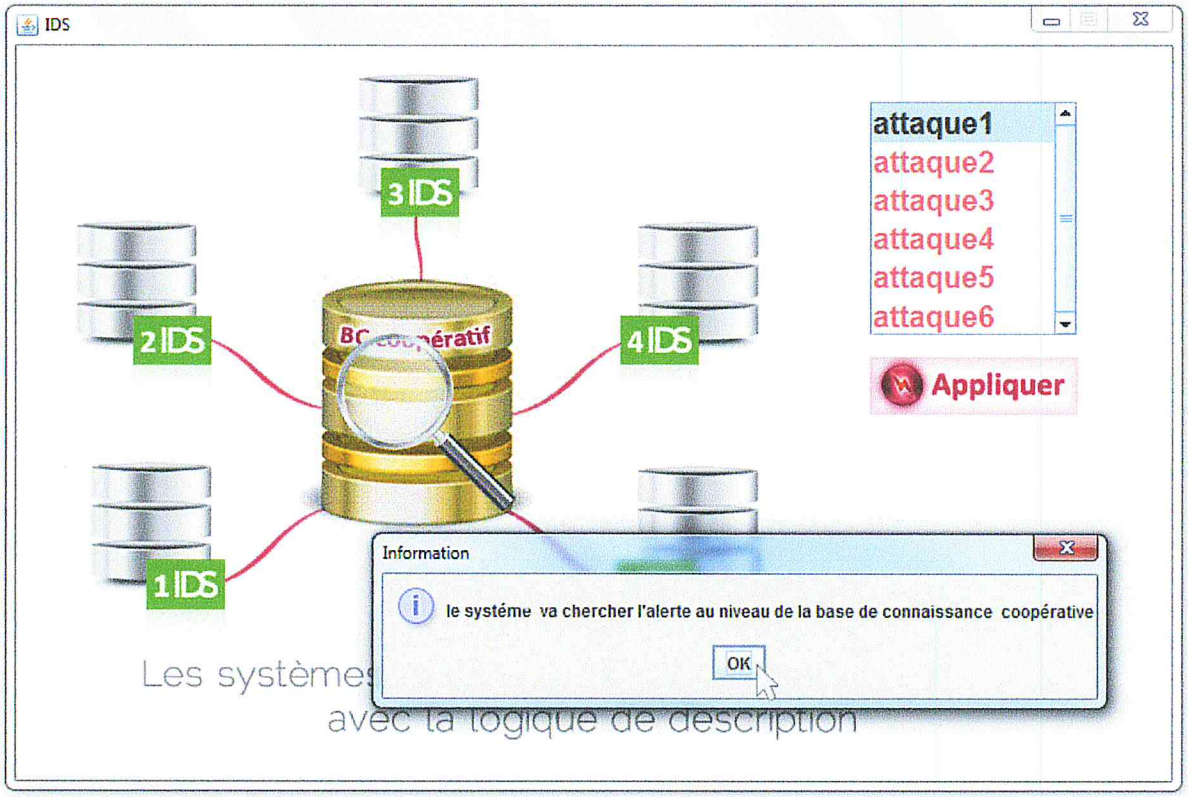


Figure 5.13: message de recherche.

Vu que l'alerte n'a pas été trouvée dans la base de connaissance de l'ids4 la recherche va commencer au niveau de la base de connaissance coopérative.

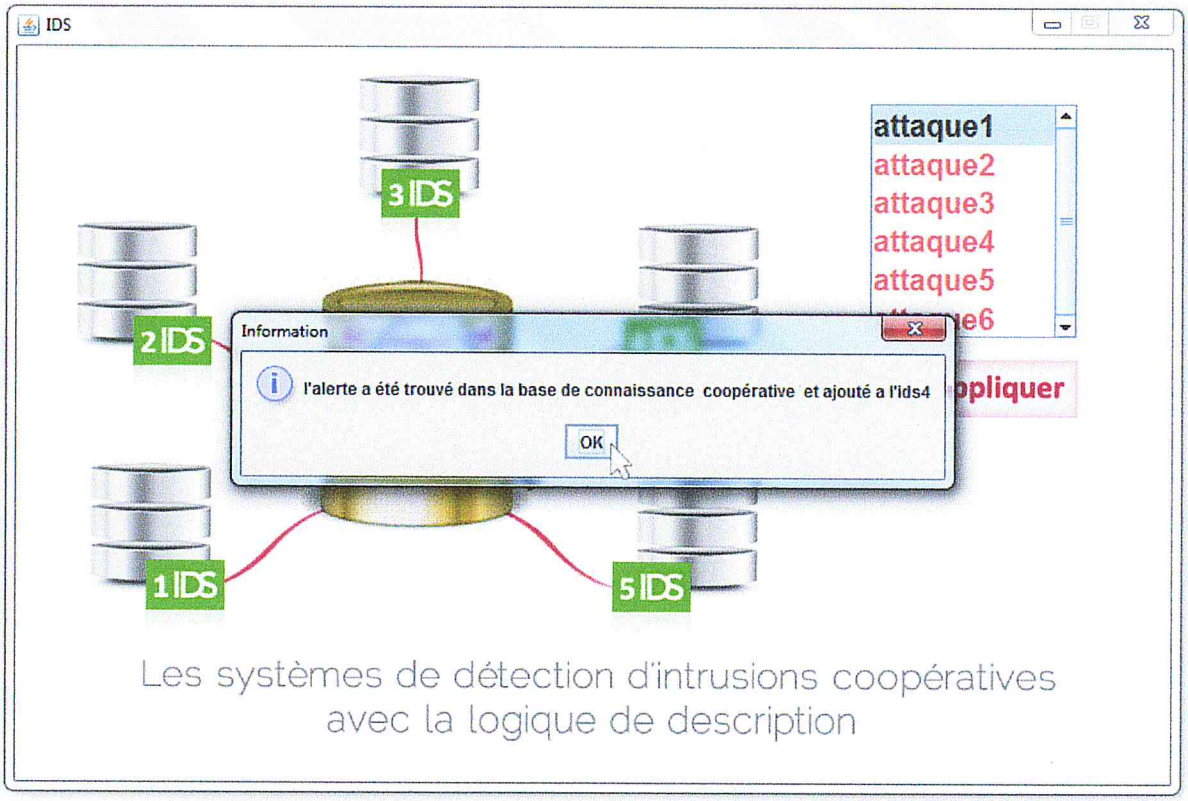


Figure 5.14: message d'information.

Un message indique que la recherche est finie et que l'alerte a été trouvée dans la base de connaissance coopérative et ajouté à l'ids4, cela indique qu'il existe une coopération entre les IDS a travers la base de connaissance coopérative.

Exemple3 :

Dans cet exemple nous illustrons le cas ou une alerte n'est pas trouver ni dans la base de connaissance de l'IDS ni dans la base de connaissance coopérative. Cela nous induira a faire des déductions pour trancher si c'est une attaque ou si c'est une fausse alerte. Les étapes sont les mêmes étapes que dans les exemples précédents.

Au niveau de la base de connaissance coopérative sa sera comme suit :

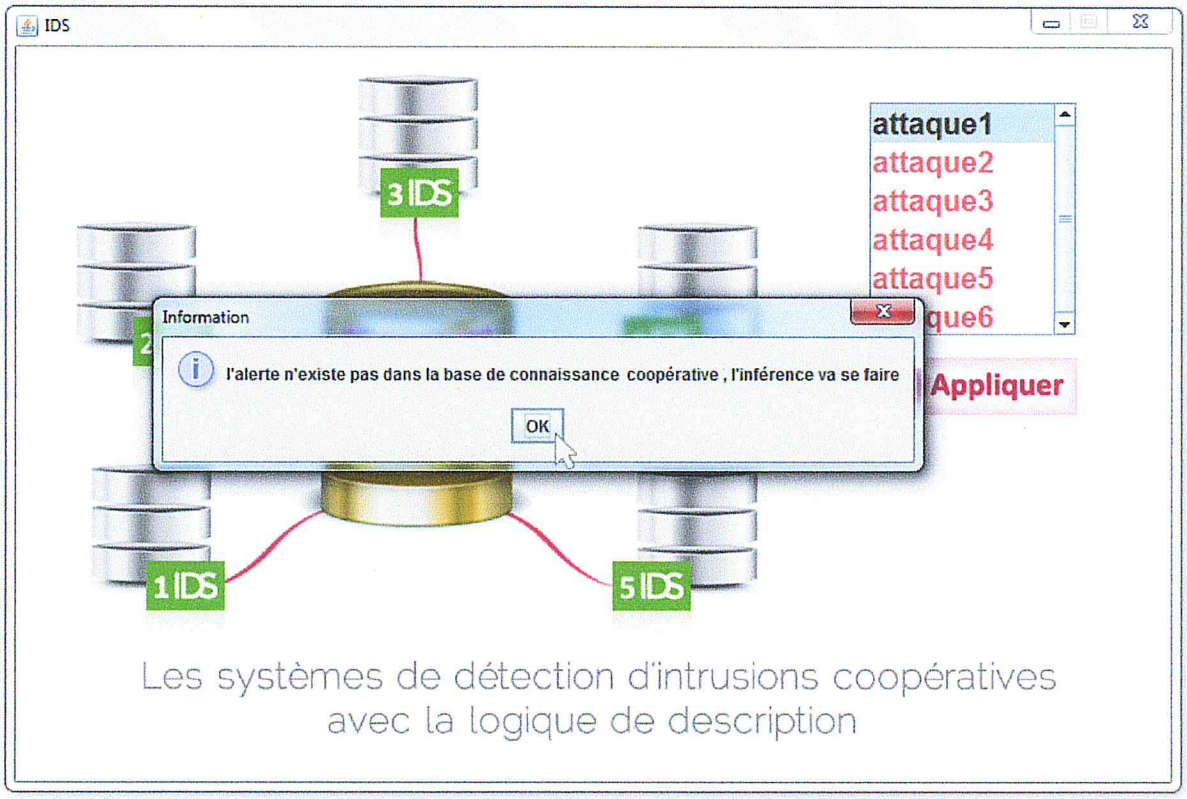


Figure 5.15: message d'information.

Dans la figure 5.15 le message nous informe que l'alerte n'a pas été trouvé dans la base de connaissance coopérative et que les déductions vont commencer pour déduire si c une attaque ou une fausse alerte.

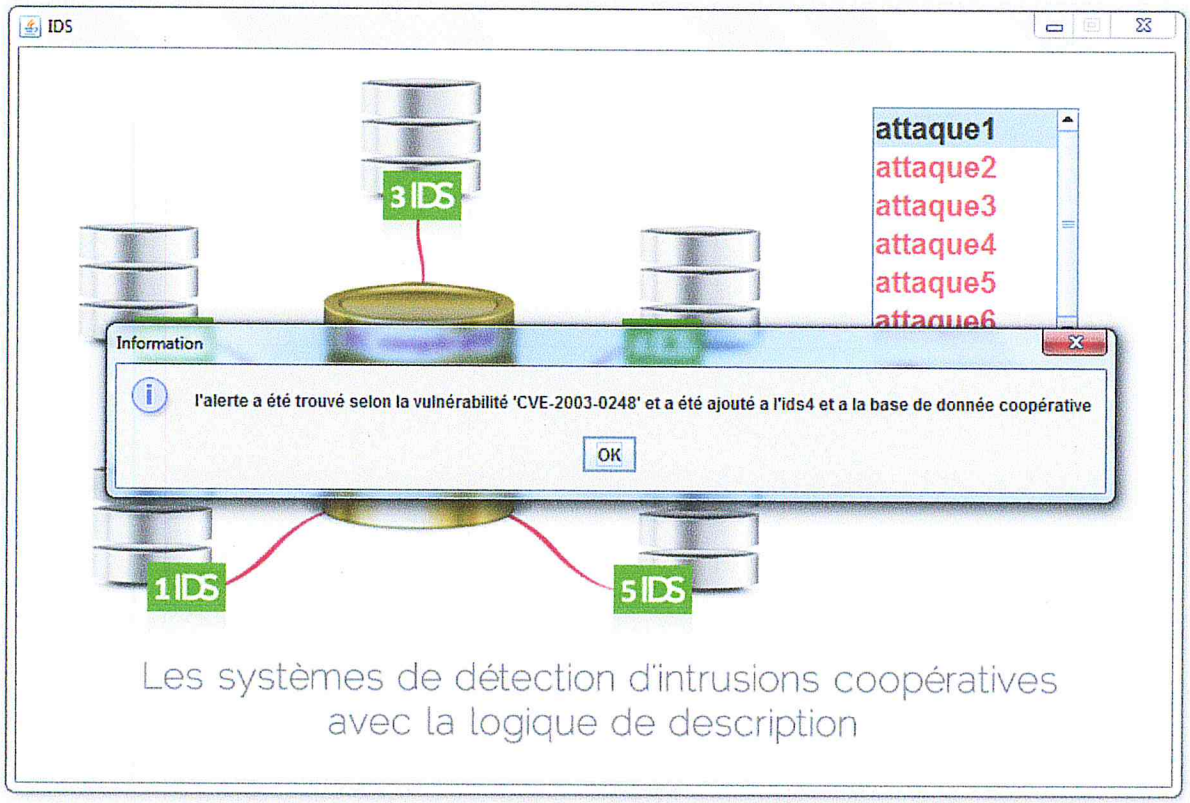


Figure 5.16: message d'information.

Dans la figure 5.16 le système indique que l'alerte est une attaque deduites des informations contenue dans la base de connaissance coopérative et cela du a une vulnérabilité du système attaqué dont le nom de la vulnérabilité CVE-2003-0248.

CONCLUSION

Nous venons de vous présenter dans ce chapitre IDS coopératif réaliser avec la logique de description, une application qui illustre une coopération entre plusieurs systèmes de détections d'intrusion ainsi que d'autres modules de sécurité tel que les analyseur et les scanners de vulnérabilité. Cela en utilisant les règles que nous avons formalisé en logique de description.

Nous avons définit ses différents composants : TBox, ABox. Nous avons doté notre application d'un mécanisme d'inférence qui permet de déduire les similitudes, la corrélation d'alerte et la diminution des fausses alertes.

Conclusion générale

Dans cette étude, nous avons proposé un système de coopération entre plusieurs IDS notamment quelques outils de sécurité, en se basant sur la théorie de récolte de données dans une base de connaissance coopérative. Le système vise à résoudre les problèmes suivants : de débordement, l'amélioration du taux de détection et les fausses alertes.

Notre première contribution dans les objectifs du travail est que par sa capacité de réduction du nombre d'alertes, le système proposé peut respectivement affaiblir le problème de saturation de l'analyste dû au très grand nombre d'alertes à analyser. L'utilisation de plusieurs systèmes de détection avec des méthodes de détection différentes améliore le taux de détection et finalement, permet une diminution de l'impact des fausses alertes par l'utilisation d'une méthode de corrélation d'alertes.

Une seconde contribution suggère une représentation formelle basée sur la logique de description des informations contextuelles en détection d'intrusions, telles les alertes générées par les IDS et les informations sur les attaques et vulnérabilités.
Cette représentation en logique de description permet aux outils de sécurité de décrire leurs observations et de raisonner sur leurs complémentarités.
Par ailleurs, cette représentation en logique de description peut être utilisée pour construire une base de connaissances contenant toutes les informations collectées, et qui sera à son tour interrogée pour la comparaison entres les attaques détectées par les IDS et les attaques contenues dans cette base de connaissance.

Enfin, une illustration de ces contributions s'est effectuée en deux étapes. Dans la première, nous avons montré l'efficacité de l'utilisation de plusieurs systèmes de détection d'intrusions reliés à une base de connaissance coopérative pour la détection des attaques, et ce en comparant les alertes décrites dans un formalisme en logique de description. Dans la deuxième, nous avons proposé et appliqué une approche dans le but de réduire le volume des fausses alertes.

Il convient de noter que l'approche coopérative proposée est simple et efficace.

Un ensemble de perspectives sont envisageables :

En premier lieu il s'agira d'inclure d'autres informations (d'autres champs) dans le formalisme d'alertes générées. Cela induira à une meilleure comparaison des attaques et l'exactitude de la similitude entre celles-ci.

En plus l'utilisation d'autres outils de détection pour la récolte d'informations dans la base de connaissance coopérative mènera à une meilleure déduction pour la diminution des fausses alertes.

Bibliographie

[1] Anderson. J. P « Computer Security Threat Monitoring and Surveillance », Technical Report James P Anderson Co., Fort Washington, PA, April 15, 1980.
[2] D. Denning «An intrusion detection models », IEEE, transaction on software engineering 13(2): 222-232, 1987.
[3] R. Heady & G. Luger & A. Maccabe & M. Servilla « The Architecture of a Network Level Intrusion Detection System ». Technical Report CS90-20, University of New Mexico, Department of Computer Science, August 1990.
[4] J. Kim « Integrating Artificial Immune Algorithms for Intrusion Detection », PhD Thesis, University College London, 2002.

[5] K. Scarfone and P. Mell. « Guide to intrusion detection and prevention systems (idps) ». Technical report, National Institute of Standards and Technology, Special Publication 800-94, USA, 2007.

[6] Przemysiam Kazienko & Piotr Dorosz « Intrusion Detection Systems (IDS) Part I -(network intrusion; attack symptoms; IDS tasks; and IDS architecture) », 2004.
http://www.windowsecurity.com/pages/article_p.asp?id=1147

[7] H. Debar & M. Dacier & A. Wespi « A revised taxonomy for Intrusion Detection Systems », Computer science, 1999.

[8] NIST, « Intrusion detection Systems ». NIST Computer Science Special reports SP 800-31, November 2001.
[9] P. Mell, V. Hu, R. Lippmann, J. Haines, and M. Zissman. «An overview of issues in testing intrusion detection systems ». Technical report, National Institute of Standards and Technology & Massachusetts Institute of Technology Lincoln Laboratory, 2003.
[10] Ph. A. Porras and A. Valdes. « Live traf_c analysis of tcp/ ip gateways. Symposium on Network and Distributed System Security (NDSS'98) » (San Diego, CA, March 98), Internet Society, 1998.
[11] H. Debar, M. Dacier, and A. Wespi. «Towards a taxonomy of intrusion detection systems ». Computer Networks, Elseiver, pages 805.822, 1999.

[12] A. Sundaram, « An introduction to Intrusion Detection », 1996
http://www.acm.org/crossroads/xrds2-4/intrus.html

[13] D. Zamboni « Intrusion Detection - Basic concept and current research at IBM », IBM Global security Analysis Lab, Information Technology Security Spring School, 2005.

[14] Mykerjee. B & Heberlein. L.T & Levitt .K.N « Network Intrusion Detection », IEEE Network, Vol 8, No 3, pp .26-41, 1994.

[15] Jackson. K & DuBois. D & Stallings. C « The NIDES Statistical Component Description and Justification » Technical Report, Computer Science Laboratory, SRI International, Menlo Park, CA, March, 1994.

[16] Teng. H & Chen . K & Lu. S « Adaptive Real-Time Anomaly Detection Using Inductively Generated Sequential Patterns », Proceeding of the 1990 Symposium on Security and Privacy, Oakland, CA, May 7-9, pp 278-284, 1990.

[17] Hervé Debar, « Application des réseaux de neurones à la détection d'intrusions sur les systèmes informatiques », thèse de doctorat, Université Paris 6, 1993.

[18] Li. Y& Wu. N & Jajosia. S & Sean Wang. X « Enhancing Profiles for Anomaly Detection Using Time Granularities » Proc. 1st ACM workshop on Intrusion Detection Systems, Athens, Greece, Nov. 2000.

[19] Steven Hofmeyr « An immunological model of distributed detection and its application to computer security ». PhD thesis, University Of New Mexico, 1999.

[20] S. Hofmeyr & S. Forrest « Architecture for an artificial immune system », Evol. Comput, vol. 8, no. 4, pp 443 – 473, 2000.

[21] Forrest .S & Perelson.A & Allen.L & Cherukuri. R «Self-Nonself Discrimination in acomputer», Proc. Of the IEEE Symposium on research in Security and Privacy, pp. 2002-212,1994.

[22] U. Aickelin & J. Greensmith & J. Twycross « Immune System Approaches to Intrusion Detection - A review », School of Computer Science, University of Nottingham, 2004.

[23] Axelsson. S. « Intrusion Detection Systems: A Taxonomy and Survey ». Technical Report No 99-15, Dept. of Computer Engineering, Chalmers University of Technology, Sweden, 2000,

[24] W. Jansen & P. Mell, T.Karygiannis & D.Marks «Mobile Agents in Intrusion Detection And Response », 2000

[25] C. Kruegel & T. Toth, « Applying Mobile Agent Technology to Intrusion Detection », Technical University of Vienna, Distributed Systems group, 2002.

[26] K. Price. « *Intrusion Detection Pages* ». Purdue University, 1998. http://www.cs.purdue.edu/coast/intrusion-detection/ids.html.

[27] B. White & E. A. Fisch, & U. W. Pooch. « Cooperating Security Managers: A Peer-Based Intrusion Detection System ». IEEE Network Journal, pp. 20-23, January/February1996.

[28] S. Staniford-Chen & S. Cheung & R. Crawford & M. Dilger & J. Frank & J. Hoagland & S. Templeton & K. Levitt & S. Walnum & C. Wee, & R. Yip. « GrIDS-A Graph-Based Intrusion Detection System for Large Network »s. Proc of the 19th National Information Systems Security Conference, 1996.

[29] S. Staniford-Chen , « GrIDS Outline Design Document ». GrIDS Project Home Page at UC Davis'sComputer Science Department, 1997. http://olympus.cs.ucdavis.edu/arpa/grids/design.html,

[30] Porras. P. A & Neumann. P. G. « EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances », Proceeding of 20th National Information System Security Conference, 1998.

[31] J. S. Balasubramaniyan & J. O. Garcia-Fernadez & D. Isacoff & E. H. Spafford& D. Zamboni. « An Architecture for Intrusion Detection using Autonomous Agents ». Technical Report Coast-TR-98-05, Computer Sciences Department, Purdue University, 1998.

[32] M. Crosbie & E. H. Spafford. « Active Defense of a Computer System using Autonomous Agents ». Technical Report CSD-TR-95-008, Purdue University, 1995.

[33] M. Crosbie & E. H. Spafford. « Defending a Computer System using Autonomous Agents ». Technical Report CSD-TR-95-022, Computer Sciences Department, Purdue University, 1995.

[34] Sattler, U., Calvanese, D. et Molitor, R. « Relationships with other formalisms. Dans Baader, F., Calvanese, D., McGuinness, D., Nardi, D. et Patel-Schneider, P. (éditeurs), The Description Logic Handbook: Theory, Implementation and Applications ». Cambridge University Press, pp. 142183, 2003.

[35] Baader, F. et Nutt, W. « Basic description logics. Dans Baader, F., Calvanese, D., McGuinness, D., Nardi, D. et Patel-Schneider, P. (éditeurs), The Description Logic Handbook : Theory, Implementation and Applications ». Cambridge University Press, pp. 47100, 2003.

[36]Baader, F., Horrocks, I. et Sattler, U. « Description logics as ontology languages for the semantic web. Dans Hutter, D. et Stephan, W. (éditeurs), Festschrift in honor of Jörg Siekmann. Lecture Notes in Articial Intelligence ». Springer-Verlag, 2003.

[37] **Nardi, D. et Brachman, R. J.** « An introduction to description logics. Dans Baader, F., Calvanese, D., McGuinness, D., Nardi, D. et Patel-Schneider, P. (éditeurs), The Description Logic Handbook : Theory, Implementation and Applications ». Cambridge University Press, pp. 544, 2003

[38] Zou, Y., Finin, T. et Chen, H., 2004. « F-owl : an inference engine for semantic web. Dans Third NASA-Goddard/IEEE Workshop on Formal Approaches to Agent-Based Systems ». Greenbelt, Maryland, 2004.

[39] Horrocks, I., Patel-Schneider, P. F. et van Harmelen, F. « From shiq and rdf to owl : The making of a web ontology language ». J. of Web Semantics 1 (1), 726, 2003

[40] Baader, F. « Appendix 1 : Description logic terminology. Dans Baader, F., Calvanese, D., McGuinness, D., Nardi, D. et Patel-Schneider, P. (éditeurs), The Description Logic Handbook : Theory, Implementation and Applications ». Cambridge University Press, pp. 495505, 2003.

[41]Horrocks, I. et Sattler, U., 2005. « A tableaux decision procedure for SHOIQ. Dans Proc. of the 19th Int. Joint Conf. on Articial Intelligence (IJCAI 2005).

[42] Donini, F. M., 2003. « Complexity of reasoning. Dans Baader, F., Calvanese, D., McGuin ness, D., Nardi, D. et Patel-Schneider, P. (éditeurs), The Description Logic Handbook : Theory, Implementation and Applications. Cambridge University Press, pp. 101141.

[43] Padadimitriou, C. H., 1994. « Computational complexity. Addison-Wesley Publishing Company, Massachusetts, É.-U.

[44] Horrocks, I. « The FaCT system. Dans de Swart, H. (éditeur), Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'98, number 1397 in Lecture Notes in Articial Intelligence ». Springer-Verlag, pp. 307312, 1998.

[45]Haarslev, V. et Möller « Description of the racer system and its applications. Dans Proceedings of the International Workshop on Description Logics (DL-2001) ». Stanford, Californie, pp. 132141, Aout 2001.

[58] Fournier-Viger, Philippe. « Un modèle de représentation des connaissances à trois niveaux de sémantique pour les systèmes tutoriels intelligents ». *Mémoire de maîtrise (M.Sc.), Université de Sherbrooke, Sherbrooke, Canada, 2005.*

[59] R.J. Brachman and J.G. Schmolze. «An overview of the Kl-One knowledge representation system ». Cognitive Science, 9(2):171_216,
1985.

[60]Schmidt-Schaub, M. et Smolka, G. « Attributive concept descriptions with complements ». Articial Intelligence 48 (1), 126, 1991.